z/OS

MVS Programming: Sysplex Services Reference

SA22-7618-03

z/OS

MVS Programming: Sysplex Services Reference

SA22-7618-03

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 1259.

Fourth Edition, September 2002

This is a major revision of SA22-7618-02.

This edition applies to Version 1 Release 4 of z/OS (5694-A01), Version 1 Release 4 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM® representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

IBM welcomes your comments. A form for readers’ comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Tables

About this document

This document supports z/OS (5694-A01) and z/OS.e (5655-G52).

This document provides the coding information, such as syntax and parameter descriptions, for the MVS services that enable multisystem applications and subsystems to:
- Run in a sysplex
- Share status information
- Send and receive messages using the XCF signalling function.
- Share data using the coupling facility.

These sysplex services can be used by authorized assembler language programs. An authorized program meets one or more of the following requirements:
- Runs in supervisor state
- Runs under PSW key 0-7
- Resides in an APF-authorized library

Who should use this document

This document is for programmers designing or modifying a multisystem application or subsystem to run in a sysplex and take advantage of the communication and data sharing functions available to sysplex members.

Programmers using this document should be extremely knowledgeable about the MVS operating system and assembler language programming.

Where to find more information

Where necessary, this document references information in other documents, using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

The following table lists the title and order number for a document related to another product.

Short title used in this document	Title	Order number
PR/SM Planning Guide	*Processor Resource/Systems Manager Planning Guide*	GA22-7123
PR/SM Planning Guide	*Processor Resource/Systems Manager Planning Guide* (S/390 processors only)	GA22-7236

| ## Information updates on the web

| For the latest information updates that have been provided in PTF cover letters and Documentation APARs
| for z/OS and z/OS.e, see the online document at:
|
| `http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ZIDOCMST/CCONTENTS`

| This document is updated weekly and lists documentation changes before they are incorporated into z/OS
| publications.

Accessing z/OS™ licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/`

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Summary of changes

Summary of changes
for SA22-7618-03
z/OS Version 1 Release 4

The book contains information previously presented in *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618-02, which supports z/OS Version 1 Release 3.

New information

- Information is added to indicate this book supports z/OS.e.
- The IXLCACHE macro provides three new request types to enhance performance when accessing a cache structure allocated in a coupling facility of CFLEVEL=12 or higher:
  - REQUEST=CASTOUT_DATALIST
  - REQUEST=CROSS_INVALLIST
  - REQUEST=WRITE_DATALIST

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this book—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our books.

Summary of changes
for SA22-7618-02
z/OS Version 1 Release 3

The book contains information previously presented in *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618-01, which supports z/OS Version 1 Release 2.

New information

- The IXCMSGO macro has new options, ALL and OTHER for the MEMBERS keyword. If MEMBERS=ALL is specified, XCF will send the message to all active members of the group. If MEMBERS=OTHER is specified, XCF will send the message to all active members of the group excluding the sender.
- An appendix with z/OS product accessibility information has been added.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7618-01
z/OS Version 1 Release 2

The book contains information previously presented in *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618-00, which supports z/OS Version 1 Release 1.

New information

- The IXCMG macro has a new option (TYPE=MEMBER) to allow the return of information for each member that resides on the local system. (APAR OW37621)

- The IXLALTER macro contains additional information about its use in system-managed duplexing rebuild.
- The following IXLCACHE request types support the BUFADDRSIZE keyword when BUFADDRTYPE=VIRTUAL:
  – REQUEST=CASTOUT_DATA
  – REQUEST=DELETE_NAMELIST
  – REQUEST=PROCESS_REFLIST
  – REQUEST=READ_COCLASS
  – REQUEST=READ_COSTATS
  – REQUEST=READ_DATA
  – REQUEST=READ_DIRINFO
  – REQUEST=UNLOCK_CASTOUT
  – REQUEST=WRITE_DATA
- The IXLCONN macro keyword SUSPEND=FAIL is updated to clarify its use during system-managed duplexing rebuild.
- The IXLLSTM macro has support for the relaxation of buffer size requirements for REQUEST=DELETE_ENTRYLIST and REQUEST=MOVE_ENTRYLIST. A new keyword, BUFINCRNUM, is also added. (APAR OW41614)
- The IXLMG macro supports the new coupling facility remote facility information mapped by IXLYAMDCFRF in IXLYAMDA.
- The IXLREBLD is updated to support system-managed duplexing rebuild.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes for SA22-7618-00 z/OS Version 1 Release 1

The book contains information also presented in *OS/390 MVS Programming: Sysplex Services Reference*

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

Symbols

The following symbols may be displayed in syntax diagrams:

Symbol	Definition
►►──	Indicates the beginning of the syntax diagram.
──►	Indicates that the syntax diagram is continued to the next line.
►──	Indicates that the syntax is continued from the previous line.
──►◄	Indicates the end of the syntax diagram.

Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

Item type	Definition
Required	Required items are displayed on the main path of the horizontal line.
Optional	Optional items are displayed below the main path of the horizontal line.
Default	Default items are displayed above the main path of the horizontal line.

Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	`►►──KEYWORD──required_item──────────────►◄`

Required choice.
 	
A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.
	`►►──KEYWORD──┬─required_choice1─┬──────►◄`
	`             └─required_choice2─┘`

Optional item.
 	
Optional items appear below the main path of the horizontal line.
	`►►──KEYWORD──┬───────────────┬─────────►◄`
	`             └─optional_item─┘`

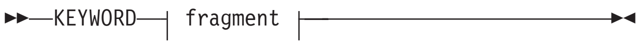
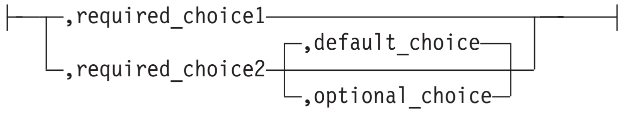
Optional choice.
 	
An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.
	`►►──KEYWORD──┬──────────────────┬──────►◄`
	`             ├─optional_choice1─┤`
	`             └─optional_choice2─┘`

Default.
 	
Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.
	`             ┌─default_choice1──┐`
	`►►──KEYWORD──┼──────────────────┼──────►◄`
	`             ├─optional_choice2─┤`
	`             └─optional_choice3─┘`

Variable. Variables appear in lowercase italics. They represent names or values.	`►►──KEYWORD──`*`variable`*`─────────────────►◄`

Repeatable item.
 	
An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.
 	
A character within the arrow means you must separate repeated items with that character.
 	
An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.
	`             ┌────────────────┐`
	`►►──KEYWORD───▼─repeatable_item─┴──────►◄`
	
	`             ┌─,──────────────┐`
	`►►──KEYWORD───▼─repeatable_item─┴──────►◄`

Table 1. Syntax examples (continued)

Item	Syntax example
Fragment. The ─┤ fragment ├─ symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.	 **fragment:**  ,required_choice1 ,required_choice2 ,default_choice ,optional_choice

Specifying a Macro Version Number

Often there is more than one version of a macro, differentiated by additional keywords or new or expanded function. For example, version 1 of the IXLCONN macro provides several new keywords in support of the structure alter function, while version 3 of the IXLCONN macro provides a new keyword to support a new resource name length attribute of a lock structure.

You can request a specific version of a macro based on the needs of your application, but you should also be attuned to the storage constraints of the installation. The version of a macro might affect the length of the parameter list generated when the macro is assembled. The size of the parameter list might grow from release to release of z/OS, perhaps affecting the amount of storage your program needs.

How to Request a Macro Version

To request a version of a macro, use the PLISTVER keyword. PLISTVER is the only parameter allowed on the list form of a macro form (MF), and it determines which parameter list the system generates. PLISTVER is optional. If you omit it, the system generates a parameter list for the lowest version that will accomodate the keywords specified. This is the IMPLIED_VERSION default.

You also have the option of coding a *specific* version number using *plistver*, or of specifying MAX.

- *plistver* allows you to code a decimal value corresponding to the version of the macro you require. The decimal value you provide determines the amount of storage allotted for the parameter list.
- MAX allows you to request that the system generate a parameter list for the highest version number currently available. The amount of storage allotted for the parameter list will depend on the level of the system on which the macro is assembled.

  IBM recommends, if your program can tolerate possible additional growth, that you always specify PLISTVER=MAX when creating the list form parameter list. MAX ensures that the list form parameter list is always long enough to hold whatever parameters might be specified on the execute form.

General Considerations When Using PLISTVER

There are some general considerations that you should keep in mind when specifying the version of a macro with PLISTVER:

1. Not all macros in z/OS have the same version numbers. The version numbers need not be contiguous.
2. If PLISTVER is omitted, the macro generates a parameter list of the **lowest** version that allows all the parameters to be processed.
3. If you code *plistver*=’n’ and then specify any version ’n+1’ keywords, the macro will not assemble.
4. If you code *plistver*=’n’ and do not specify any version ’n’ keywords, the macro will generate a version ’n’ parameter list.

Each macro in *z/OS MVS Programming: Sysplex Services Reference* that has one or more versions contains the PLISTVER keyword in the syntax diagram and in the parameter descriptions. At the beginning of each such macro description, there is a topic entitled “Understanding *macname* Version Support”, where *macname* is the macro name. That topic specifies the range of values for *plistver* and lists the keywords and functions applicable for each version of the macro.

IHABLDP — Build Dump Parameter List Service

Description

The IHABLDP macro builds the STRLIST parameter list that is specified as input on the SDUMPX macro when you request coupling facility structure information. IHABLDP builds the STRLIST parameter list in a block of storage that you provide to the macro. The STRLIST parameter list is mapped by the macro IHASDSTR. The IHABLDP macro does not initiate a dump request.

Each time you invoke IHABLDP, you specify a TYPE parameter to indicate which operation is to be performed on the STRLIST parameter list. The TYPE option determines which entry is to be built in the STRLIST parameter list.

- TYPE=INITIAL initializes the storage to binary zeros and generates an initialized header for the dump parameter list. The length of the dump parameter list header is 24 bytes.
- TYPE=STRUCTURE generates a structure entry in the dump parameter list. The length of this entry is 48 bytes.
- TYPE=STRRNG generates a structure range entry in the dump parameter list. The length of this entry is 12 bytes.
- TYPE=STROPT generates a structure option entry in the dump parameter list. The length of this entry is 12 bytes.
- TYPE=ENDLIST completes the construction of the dump parameter list. No entries will be added to the dump parameter list.

The data is dumped in the order in which you requested it. Serialized ranges are dumped before unserialized ranges.

The amount of storage required for the STRLIST parameter list depends on the number and type of entries in the parameter list. You determine the size by adding the number of bytes for each entry, as described above.

- The minimum STRLIST parameter list consists of a header and one structure entry (TYPE=STRUCTURE); thus the minimum size is 72 bytes.
- The maximum STRLIST parameter list can consist of 47 structures and 6 ranges. For each structure less than 47, you can specify an additional 10 ranges. For example:

  ```
  47 structures and  6 ranges
  46 structures and 16 ranges
  44 structures and 36 ranges
  ```

  If you specify more than the maximum allowed in the STRLIST, the system truncates the extraneous entries and indicates that the dump is a partial one by setting the SDRSTRLE flag in IHASDRSN, SDUMP Partial Reason Codes.

IHABLDP processes requests with incorrect parameter list lengths in the following ways:

1. Minimum storage not specified at initialization

   If less than 72 bytes of storage is specified when TYPE=INITIAL is requested, IHABLDP returns a return code of 8 and a reason code of 4 to indicate that insufficient space is available. Subsequent IHABLDP invocations that reference the same STRLIST parameter list also fail.
2. Insufficient storage not specified after initialization

   If sufficient storage is not available in an initialized STRLIST parameter list to add another entry, IHABLDP returns a return code of 8 and a reason code of 4. Subsequent IHABLDP invocations that reference the same STRLIST parameter list also are unable to add entries to the parameter list and receive the same return and reason codes. When you invoke TYPE=ENDLIST to complete the construction of this parameter list, IHABLDP sets the total length of the list to 16 bytes less than the block of storage that you initially provided to the macro. The SDUMPX macro must be responsible to check that all entries in the parameter list are processed.

To use the macro, follow these guidelines:

- Specify TYPE=INITIAL before any other invocation of IHABLDP.
- Specify TYPE=STRUCTURE for each structure desired in the dump before specifying TYPE=STRRNG or TYPE=STROPT for that structure.

  Once TYPE=STRUCTURE is processed, you can specify one or more TYPE=STRRNG or TYPE=STROPT options until all the desired ranges for a requested structure are specified. IHABLDP allows you prioritization of information that you request for a specified structure. You can specify the STRRNG and STROPT options in any order to give you more flexibility in requesting the most information at the earliest time.
- On TYPE=STRRNG, the starting range value must not be greater than the ending range value. If the starting value is greater, IHABLDP does not allow the structure range entry to be added to the parameter list and returns a return code of 8 and a reason code of 8.
- Specify TYPE=ENDLIST to complete the building of the parameter list. Make sure that you have specified all of the requested structures and their requested ranges.
- Do not attempt to change the built parameter list once you have specified TYPE=ENDLIST.

Environment

Minimum authorization:	One of the following: 1. Problem or supervisor state 2. Any PSW key
Dispatchable unit mode:	Task or SRB mode
Amode:	31-bit
ASC mode:	Primary or access register
Locks:	No locks held
Interrupt status:	Enabled or disabled for I/O and external interrupts
Control parameters:	Control parameters must be in the primary address space or be in an address space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before invoking IHABLDP. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

None.

Input Register Information

Before issuing the IHABLDP macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IHABLDP macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code if GPR15 return code is non-zero
1	Used as work register by the system
2 - 13	Unchanged
14	Used as work register by the system
15	Return code

The access registers contain:

Register	**Contents**
0 - 1	Used as work registers by the system
2 - 13	Unchanged
14 -15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Performance Implications

None.

Understanding IHABLDP Version Support

The IHABLDP macro supports versions in the range of 0 - 1.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IHABLDP macro.
- The following keywords and functions are supported by all versions starting with version 1 and higher of the IHABLDP macro.
  OBJECT=EMCONTROLS (only for keyed list structures allocated in a coupling facility with CFLEVEL=4 or higher)
  OPTION=EVENTQS (only for keyed list structures allocated in a coupling facility with CFLEVEL=4 or higher)

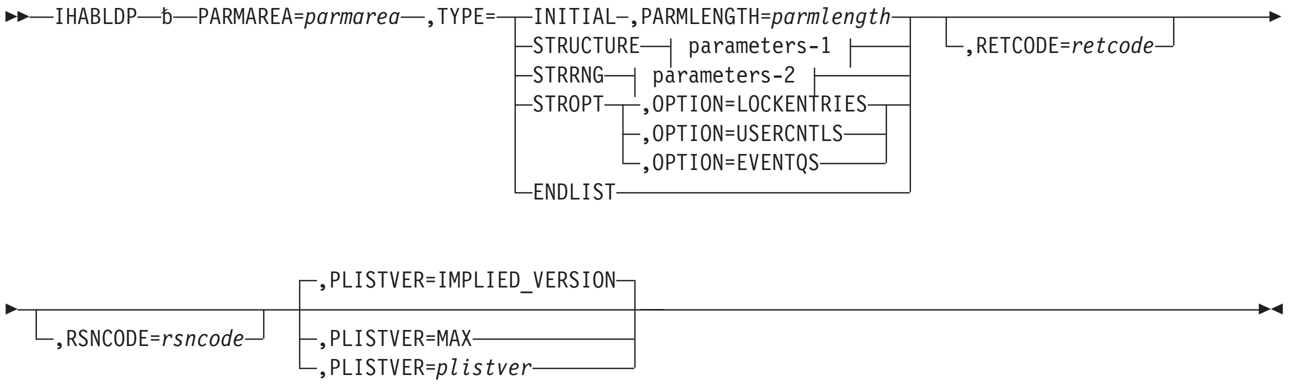
Specify the version of the parameter list that you want generated with the PLISTVER keyword. See "Specifying a Macro Version Number" on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax

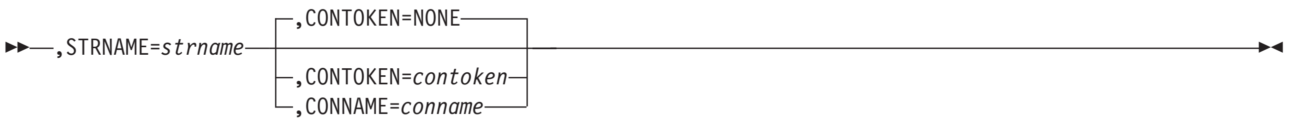
The syntax of the IHABLDP macro is written as follows:

IHABLDP Macro

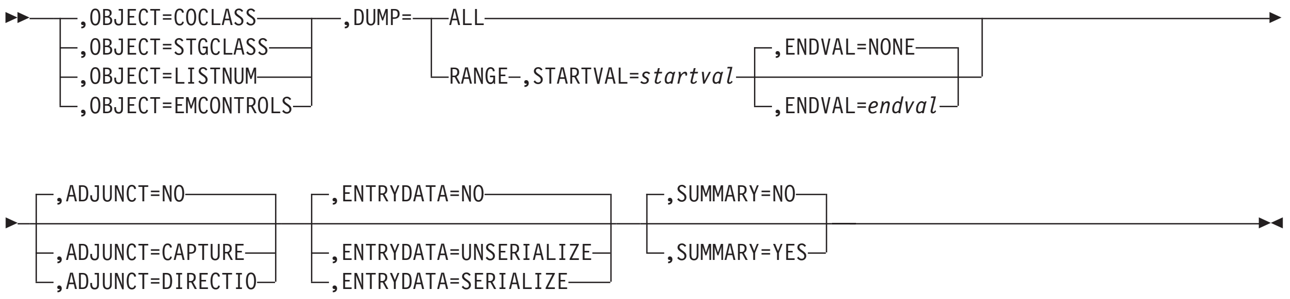
main diagram



parameters-1



parameters-2



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ADJUNCT=<u>NO</u>
,ADJUNCT=CAPTURE
,ADJUNCT=DIRECTIO

Use this input parameter to specify whether or not the system should include adjunct data in the dump for all entries within the specified range. If the structure contains adjunct data, ADJUNCT specifies whether the adjunct data should be dumped.

NO
Do not dump adjunct data.

CAPTURE
While serialization is held, capture the adjunct data along with the entry controls.

DIRECTIO
After serialization is released, dump the adjunct data via direct I/O to the dump data set after the controls are captured.

,CONNAME=*conname*

Use this input parameter to specify the name of the connected user. When this keyword is specified for a cache structure, the VECTORINDEX with which the indicated user has registered interest in each entry that is dumped, will be included in the dump along with the directory entry information. When this keyword is specified for a list structure, it is ignored.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the name of the connected user.

,CONTOKEN=NONE
,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The CONTOKEN uniquely identifies a user's connection to a structure. When this keyword is specified for a cache structure, the VECTORINDEX with which the indicated user has registered interest in each entry that is dumped, will be included in the dump along with the directory entry information. When this keyword is specified for a list structure, it is ignored.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the connect token returned by the IXLCONN service.

,DUMP=ALL
,DUMP=RANGE
Use this input parameter to specify the ranges to be dumped of a specified object.

ALL
All values of the specified object are to be dumped. If you also specify SUMMARY=NO (the default), the system dumps all the object controls and their associated entry controls. If you specify SUMMARY=YES, then only the object controls are dumped. For example:

- If you specify OBJECT=COCLASS with DUMP=ALL and SUMMARY=NO, the system dumps all of the object controls and their associated entries in all of the cast-out classes.
- If you specify OBJECT=COCLASS with DUMP=ALL and SUMMARY=YES, the system dumps only the object controls for all cast-out classes.

RANGE
A range of values of the specified object are to be dumped. If you also specify SUMMARY=NO (the default), the system dumps all the object controls and their associated entry controls. For example:

- If you specify OBJECT=COCLASS with DUMP=RANGE and SUMMARY=NO, the system dumps the object controls for the range of cast-out classes and also the cast-out classes' entry controls.
- If you specify OBJECT=COCLASS with DUMP=RANGE and SUMMARY=YES, the system dumps all of the object controls for the range of cast-out classes and no entries for any of the cast-out classes.

,ENDVAL=NONE
,ENDVAL=*endval*
Use this input parameter to specify the ending range value. If you do not specify ENDVAL or if you specify ENDVAL=NONE, then only the value specified for STARTVAL is dumped for this range.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the ending range value.

,ENTRYDATA=NO
,ENTRYDATA=UNSERIALIZE
,ENTRYDATA=SERIALIZE
Use this input parameter to specify whether or not entry data should be included in the dump for all the entries in the specified range.

NO
Do not dump the entry data.

UNSERIALIZE
Release structure dump serialization before writing entry data to the dump data set.

SERIALIZE
Keep structure dump serialization until after the entry data has been written to the dump data set.

,OBJECT=COCLASS
,OBJECT=STGCLASS
,OBJECT=LISTNUM
,OBJECT=EMCONTROLS
Use this input parameter to specify what type range should be dumped.

COCLASS
The requested range is a range of cast-out classes. Use COCLASS only if the requested structure is a cache structure. If the requested structure is not a cache structure, the system dumps nothing for this entry.

STGCLASS
The requested range is a range of storage classes. Use STGCLASS only if the requested structure is a cache structure. If the requested structure is not a cache structure, the system dumps nothing for this entry.

Note: If a data entry appears in both a castout class and a storage class and you request that both classes are to be dumped in two IHABLDP parameter list entries, the system dumps the entry twice.

LISTNUM
The requested range is a range of list numbers. Use LISTNUM only if the requested structure is a list structure. If the requested structure is not a list structure, the system dumps nothing for this entry.

EMCONTROLS
The requested range is a range of list numbers for which event monitor controls should be dumped. Use EMCONTROLS only if the requested structure is a keyed list structure allocated in a coupling facility with CFLEVEL=4 or higher. If the requested structure is not such a list structure, the system dumps nothing for this entry.

,OPTION=LOCKENTRIES
,OPTION=USERCNTLS
,OPTION=EVENTQS
Use this input parameter to specify the option you want in the dump.

LOCKENTRIES
Include the lock table entries associated with the requested structure in the dump. Use LOCKENTRIES only when the requested structure is a list structure.

USERCNTLS
Include the user attach controls in the dump.

EVENTQS
Include in the dump the user event queues for a list structure allocated in a coupling facility with CFLEVEL=4 or higher.

PARMAREA=*parmarea*
Use this output parameter to specify the name of the storage area that is to be used for building the dump parameter list. This storage area eventually will be passed as input to the SDUMPX macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the storage area that is to be used for building the dump parameter list.

,PARMLENGTH=*parmlength*
Use this input parameter to specify the length of the storage area used for the dump parameter list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field containing the length, **in bytes**, of the storage area used for the dump parameter list.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IHABLDP Version Support" on page 9 for a description of the options available with PLISTVER.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the reason code.

,STARTVAL=*startval*
Use this input parameter to specify the starting range value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the starting range value.

,STRNAME=*strname*
Use this input parameter to specify the name of the cache or list structure for which information is being requested.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the name of the cache or list structure for which information is being requested.

,SUMMARY=NO
,SUMMARY=YES
Use this input parameter to specify whether or not the dump should include only a summary of the object requested. If SUMMARY=YES, the entry control information is not included; only structure and class information is included in the dump.

NO
Do not do a summary dump of the requested object; include entry controls.

YES
Do a summary dump of the requested object; do not include entry controls.

,TYPE=INITIAL
,TYPE=STRUCTURE
,TYPE=STRRNG
,TYPE=STROPT
,TYPE=ENDLIST
Use this input parameter to specify the type of operation to be performed on the dump parameter list.

INITIAL
Initialize the storage to binary zeros and generate an initialized header for the STRLIST dump parameter list. The header for the dump parameter list is 24 bytes long.

STRUCTURE
Generate a structure entry in the STRLIST dump parameter list. The structure entry is 48 bytes long.

STRRNG
Generate a structure range entry in the STRLIST dump parameter list. The structure range entry is 12 bytes long.

STROPT
Generate a structure option entry in the STRLIST dump parameter list. The structure option entry is 12 bytes long.

ENDLIST
End building the STRLIST dump parameter list. You must specify ENDLIST when you complete building the STRLIST dump parameter list.

ABEND Codes

None.

Return and Reason Codes

When the system returns control to the caller, GPR 15 (and *retcode*, if you coded RETCODE) contains the return code and GPR 0 (and *rsncode*, if you coded RSNCODE) contains the reason code.

Table 2. Return and Reason Codes for the IHABLDP Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
00	None	**Meaning:** IHABLDP completed successfully. **Action:** None
08	04	**Meaning:** There is insufficient space in the dump parameter list to add the requested entry. **Action:** Ensure that you have correctly calculated the amount of storage required for the STRLIST parameter list.
08	08	**Meaning:** The range entry was not added to the dump parameter list because the starting range value was greater than the ending range value. **Action:** Verify that you specified the correct starting and ending range values on the TYPE=STRRNG request.

IXCARM — Request Automatic Restart Management Services

Description

Use the IXCARM macro to request the following services from the automatic restart management function of XCF:

- Register as an element of the automatic restart manager (REGISTER parameter)
- Wait until predecessor elements have been restarted, if applicable (WAITPRED parameter)
- Mark an element as ready to accept work (READY parameter)
- Deregister from the automatic restart manager (DEREGISTER parameter)
- Associate an element with another element (ASSOCIATE parameter).

For more information about the services performed by the IXCARM macro, see *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Minimum authorization:	• Supervisor state or PKM allowing key 0 - 7 • Problem state and non-PKM system key is supported for all request types if the caller has SAF authority. SAF authorization to the entity IXCARM.elemtype.elementname is required. If elemtype is not specified, then the entity defaults to IXCARM.DEFAULT.elementname.
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry in the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCARM. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

The IXCYARAA mapping macro provides the format of the area pointed to by the ANSAREA parameter. If you intend to use that area, include the IXCYARAA mapping macro in your program.

The areas specified by STARTTXT, EVENTEXITPL, and ANSAREA must be in the primary address space or be in an address/data space that is addressable through a public entry in the caller's dispatchable unit access list (DU-AL).

Include the IXCYARM mapping macro in your program. This macro provides a list of equate symbols for users of IXCARM.

Initialize the area that contains the event exit parameter list (EVENTEXITPL) before issuing the IXCARM macro with the REQUEST=REGISTER parameter.

Restrictions

The caller cannot have any enabled, unlocked task (EUT) FRRs established.

In general, all request types must be issued from the same home address space from which the IXCARM REQUEST=REGISTER request was issued. However, IXCARM requests for elements that represent abstract resources (ELEMBIND=CURSYS is specified) can be issued from any address space. IXCARM DEREGISTER requests for elements that represent jobs or started tasks (ELEMBIND=CURJOB is specified) can be issued from the master address space or the same home address space from which the REGISTER request was issued.

The security administrator may control the use of automatic restart management by unauthorized applications through the use of RACF or another security product. Unauthorized applications can use only element names that are registered as ELEMBIND=CURJOB elements. Ensure that you are authorized to issue the IXCARM macro for such unauthorized applications.

To define profiles that control unauthorized applications' use of automatic restart management, the security administrator can:

1. Define resource profile IXCARM.elemtype.elemname in the FACILITY class.
2. Specify the users who have access to automatic resource management services using the RACF PERMIT command.
3. Make sure the FACILITY class is active and generic profile checking is in effect. If in-storage profiles are maintained for the FACILITY class, refresh them.

For example, if a user wants to permit an unauthorized application with an *elemtype* of xxxxxxxx and an *elemname* of yyyyyyyyyyyyyyyy to use automatic restart management services, the security administrator can use the following commands:

```
RDEFINE FACILITY IXCARM.xxxxxxxx.yyyyyyyyyyyyyyyy UACC(NONE)

PERMIT IXCARM.xxxxxxxx.yyyyyyyyyyyyyyyy CLASS(FACILITY)
       ID(userid) ACCESS(UPDATE)

SETROPTS CLASSACT(FACILITY)
```

For information about RACF, see *z/OS Security Server RACF Security Administrator's Guide*.

Input Register Information

Before issuing the IXCARM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	**Contents**
0	If GPR 15 contains a 0, GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-14	Unchanged.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-14	Unchanged
15	Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Understanding IXCARM Version Support

The IXCARM macro supports version 1 keywords and functions.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax

The syntax of the IXCARM macro is as follows:

IXCARM macro

```
►►──IXCARM──␢──,REQUEST=──┬─REGISTER──┤ parameters-1 ├──────────────────────────┬──────────►
                          │           ┌─,RMTOKEN=NO_RMTOKEN─┐                   │
                          ├─WAITPRED──┴─,RMTOKEN=rmtoken────┴───────────────────┤
                          │        ┌─,RMTOKEN=NO_RMTOKEN─┐                      │
                          ├─READY──┴─,RMTOKEN=rmtoken────┴──────────────────────┤
                          │             ┌─,RMTOKEN=NO_RMTOKEN─┐                 │
                          ├─DEREGISTER──┴─,RMTOKEN=rmtoken────┴─────────────────┤
                          │            ┌─,RMTOKEN=NO_RMTOKEN─┐                  │
                          └─ASSOCIATE──┴─,RMTOKEN=rmtoken────┴──,TELEMENT=telement─┘

                                                    ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬───────┼───────────────────────────┼──────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘       ├─,PLISTVER=MAX─────────────┤
                                                    └─,PLISTVER=plistver────────┘

   ┌─,MF=S────────────────────────────┐
►──┼──────────────────────────────────┼────────────────────────────────────────────────►◄
   │                   ┌─,0D─────┐    │
   ├─,MF=(L─,mfctrl────┴─,mfattr─┴─)──┤
   │                   ┌─,COMPLETE─┐  │
   └─,MF=(E─,mfctrl────┴─,COMPLETE─┴─)┘
```

parameters-1

```
                                  ┌─,EVENTEXIT=NO_EVENTEXIT──────────────────────────────────────────────────┐
►►──,ELEMENT=element──────────────┼───────────────────────────────────────────────────────────────────────────┼──►
                                  │                            ┌─,EVENTEXITPL=NO_EVENTEXITPL────────────────┐ │
                                  └─,EVENTEXIT=eventexit───────┼────────────────────────────────────────────┼─┘
                                                               └─,EVENTEXITPL=eventexitpl──,EXITPLLEN=exitpllen┘

   ┌─,STARTTXT=NO_STARTTXT──────────────────────────┐  ┌─,ELEMTYPE=NO_ELEMTYPE─┐
►──┼────────────────────────────────────────────────┼──┼───────────────────────┼──────────────────────────────►
   └─,STARTTXT=starttxt──,STARTTXTLEN=starttxtlen───┘  └─,ELEMTYPE=elemtype────┘

                                                       ┌─,RMTOKEN=NO_RMTOKEN─┐
                              ┌─,ELEMBIND=CURJOB───────┼─────────────────────┼──┐
   ┌─,TERMTYPE=ALLTERM──┐     │                        └─,RMTOKEN=rmtoken────┘  │   ┌─,RESTARTTIMEOUT=NORM─┐
►──┼────────────────────┼─────┼─────────────────────────────────────────────────┼───┼──────────────────────┼───►
   ├─,TERMTYPE=ELEMTERM─┤     └─,ELEMBIND=CURSYS─,RMTOKEN=rmtoken───────────────┘   └─,RESTARTTIMEOUT=LONG─┘
   └─,TERMTYPE=SYSTERM──┘

►─,ANSAREA=ansarea──────────────────────────────────────────────────────────────────────────────────────────►◄
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined.

REQUEST=REGISTER
REQUEST=WAITPRED
REQUEST=READY
REQUEST=DEREGISTER
REQUEST=ASSOCIATE

Use this input parameter to specify the type of request:

REGISTER

Requests registration (or re-registration) of a user with automatic restart manager. The user can be a batch job, started task, or an abstract resource that automatic restart manager is to restart when the work fails or the system on which the work is currently registered fails.

WAITPRED

Requests that automatic restart manager suspend the processing of the issuing program during restart processing for the element until any predecessor elements in the same restart group that are also being restarted have indicated that they are ready. The predecessors to the element that has issued this WAITPRED request are identified from “level” specifications in the active automatic restart management policy.

READY

Requests that automatic restart manager mark a registered user as READY to accept work. If a WAITPRED request has not previously been issued by the user, the READY request will perform an implicit WAITPRED to wait for predecessor elements to initialize. The system will suspend the current task until all predecessor elements have indicated that they are ready to accept work.

DEREGISTER

Requests deregistration of a registered user. The DEREGISTER request must be issued from the same address space that issued the REGISTER request, with the following exceptions:

- If the REGISTER request specified ELEMBIND=CURSYS, the DEREGISTER may be done from any address space.
- Any element may be DEREGISTERed from an address space termination resource manager running in the MASTER address space, if the RMTOKEN returned on the REGISTER is supplied. The application should only issue this request from the master address space while running under a resource manager. If the address space under which the REGISTER request was issued is not terminating and the application issues any IXCARM requests from that address space after a DEREGISTER from the master address space, results are unpredictable.

ASSOCIATE

Requests that a registered user be associated with another element for takeover or restart processing purposes. This “association” suppresses all automatic restart manager restarts of the element identified in the TELEMENT parameter of this macro. The element issuing the ASSOCIATE request must be in the “available” state (that is, have issued the IXCARM REQUEST=READY macro).

If the issuer of the ASSOCIATE request terminates or deregisters from automatic restart manager, then the association is broken and the element specified in TELEMENT again becomes eligible for automatic restart manager restarts.

,ANSAREA=*ansarea*

Use this output area to specify an answer area to contain registration information on return from an IXCARM REQUEST=REGISTER request. This area must be 32 bytes long. Its format is described in the IXCYARAA mapping macro. Data returned in this area is valid only upon successful completion of this request (return code of 0 or 4), and if the validity flag is on (field ARAAREGTYPE is not zero).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte area where the system will put the registration information.

,ELEMBIND=CURJOB
,ELEMBIND=CURSYS

Use this input parameter to specify the relationship between the element and the system. The element bind specifies the minimum bind that must be broken for automatic restart management to take action for the element. The ELEMBIND keyword indicates the type of resource the element represents.

CURJOB

The element has a minimum bind to the batch job or started task under which the element is registering. Specify ELEMBIND=CURJOB if the batch job or started task represented by the element needs to be restarted if it fails. The IXCARM request must be issued under a batch job or started task. There can be only one automatic restart management element registered with ELEMBIND=CURJOB per batch job or started task.

CURSYS

The element has a minimum bind to the system on which the element is registering. The element represents an abstract resource. An abstract resource is a program or a set of programs that is only associated with (or has a bind to) the system on which it is running. Specify ELEMBIND=CURSYS if the application registering with automatic restart management needs to be restarted only when the system fails. No address space, batch job, or started task failure will cause the element to be restarted. A REGISTER request that specifies ELEMBIND=CURSYS can be issued from any address space.

There can be more than one automatic restart management element registered with ELEMBIND=CURSYS per batch job, started task, or address space because the element is not associated with any of these units of work. There is no persistent restart text to restart the element. Therefore, the text of the command to restart this element must be specified either by the application when it registers, in the automatic restart management policy, or by an installation-written element restart exit. If the system fails and restart command text is not provided, the element will be deregistered.

ELEMBIND=CURSYS cannot be specified with TERMTYPE=ELEMTERM.

,ELEMENT=*element*

Use this input parameter to specify the name of the element that the automatic restart manager is to register. The rules for specifying an element name are:

- Valid characters are:
  - Uppercase alphabetic characters
  - The numbers 0 through 9
  - $, #, @, and underscore (_).
- The first character may not be a number.
- The name must be 16 characters long, padded on the right with blanks
- The name must be unique across the sysplex

Element names that start with A through I and SYS are reserved for use by IBM.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the element.

,ELEMTYPE=NO_ELEMTYPE
,ELEMTYPE=*elemtype*

Use this input parameter to specify the type of element to be associated with this user. The element type is used:

- To facilitate the assigning of a “level” that is used to sequence the restarts of multiple elements during automatic restart management restarts after a system failure.
- To communicate the element type to listeners of the automatic restart management ENF Code 38 for events that pertain to this element.

The rules for specifying an element type are:

- Valid characters are:
  - Uppercase alphabetic characters
  - The numbers 0 through 9
  - $, #, and @
- The first character may not be a number
- The type must be 8 characters long, padded on the right with blanks
- The element type does not have to be unique.

Element types that start with A through I and SYS are reserved for use by IBM. See *z/OS MVS Programming: Sysplex Services Guide* for a list of the assigned ELEMTYPE values.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the type of element being registered.

,EVENTEXIT=NO_EVENTEXIT
,EVENTEXIT=*eventexit*

Use this input parameter to specify the name of the event exit routine that is to be given control when certain events occur to the element. The event exit can be used to perform specialized processing for the element.

The event exit must be contained in the authorized linklib concatenation or LPA (not steplib or tasklib) of both the system where the REGISTER request was invoked and any potiential target restart systems. The event exit will be invoked vith a BALR/BAKR sequence of instructions.

The event exit name must be 8 characters long, padded on the right with blanks.

An unauthorized application cannot specify an event exit routine when registering with automatic restart management.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the name of the event exit routine.

,EVENTEXITPL=NO_EVENTEXITPL
,EVENTEXITPL=*eventexitpl*

Use this input parameter to specify the name of the event exit parameter list. If the IXCARM invocation is an AR ASC mode, this parameter list can be in either the primary address space or in a address/data space that is addressable through a public entry on the caller's DU-AL.

Automatic restart management makes a copy of this parameter list to pass to the event exit. The parameter list should not contain data that is MVS-image dependent (such as addresses) because the exit may run on another MVS image (an automatic restart manager may restart the element on another image).

If this parameter is specified, you must specify a value for EXITPLLEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list to be used by the event exit.

,EXITPLLEN=*exitpllen*

Use this input parameter to specify the length of the event exit parameter list. The parameter list may be from 0 through 255 bytes in length. This parameter is required when EVENTEXITPL is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the length of the event exit parameter list.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXCARM Version Support" on page 17 for a description of the options available with PLISTVER.

,RESTARTTIMEOUT=NORM
,RESTARTTIMEOUT=LONG
Use this input parameter to indicate how long automatic restart management should wait for a restarted element to re-register. This parameter is applicable only when the element's restart timeout is determined by automatic restart manager's default value. If the installation's active ARM policy specifies a RESTART_TIMEOUT value for this element, the RESTARTTIMEOUT specification on a REGISTER request is ignored.

NORM
Automatic restart management is to wait 5 minutes (the normal time-out value) for this restarted element to issue the IXCARM REQUEST=REGISTER request.

LONG
Automatic restart management is to wait up to 6 hours (the long time-out value) for this restarted element to issue the IXCARM REQUEST=REGISTER request.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RMTOKEN=NO_RMTOKEN
,RMTOKEN=*rmtoken*
Use this input and output parameter to specify the 16-byte field that contains a restart manager token.

- The restart manager token is returned by an IXCARM REQUEST=REGISTER request. The contents of the restart token must not be modified by the application registering the element.
- When registering with ELEMBIND=CURJOB, applications can specify the restart manager token on subsequent REQUEST=DEREGISTER requests issued from the master address space. Applications cannot specify the restart manager token on READY, WAITPRED, ASSOCIATE, or DEREGISTER requests from the address space in which the element is registered.
- When registering with ELEMBIND=CURSYS, applications must specify the restart manager token on subsequent READY, WAITPRED, ASSOCIATE, and DEREGISTER requests to identify the element. The applications can specify the restart manager token on these requests from any address space.

If the IXCARM invocation is in AR ASC mode, the area containing the restart manager token can be in either the primary address space or in an address or data space that is addressable through a public entry on the caller's DU-AL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field containing the restart manager token.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTTXT=NO_STARTTXT
,STARTTXT=*starttxt*
Use this input parameter to specify the text of the command to be used to restart this element. When command text is specified, STARTTXTLEN must contain the length of the command text.

If the IXCARM invocation is in AR ASC mode, the text can be in either the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.

If this parameter is specified, it will take precedence over the automatic restart manager's reuse of the START command that most recently started this element (persistent restart text). A command specified with the STARTXT parameter can be overridden by a RESTART_METHOD specified for the element in the active ARM policy, unless the RESTART_METHOD defaults to or specifies PERSIST.

The STARTXT pafameter is valid only for started tasks. If specified with a REGISTER request from a batch job, the registration request will be rejected.

Notes:

1. This parameter is valid only for started tasks.
2. A command provided through the STARTTXT parameter cannot contain symbolic substitution parameters (such as &SYSNAME).
3. An unauthorized application cannot specify the STARTTXT parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the text of the command to be used to restart this element.

,STARTTXTLEN=*starttxtlen*
Use this input parameter to specify the length of the command specified in the STARTTXT parameter. This command may be from 0 through 126 bytes in length.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the length of the command text.

,TELEMENT=*telement*
Use this input parameter to specify the name of the element that should be associated with the element issuing this IXCARM request. The element name must be 16 characters long, padded on the right with blanks. This name must meet all of the requirements described under the ELEMENT parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the element to be associated with the element issuing the IXCARM macro.

,TERMTYPE=ALLTERM
,TERMTYPE=ELEMTERM
,TERMTYPE=SYSTERM
Use this input parameter to specify the type of termination for which this element may be restarted. A specification of TERMTYPE for this element in the active ARM policy will override this keyword, except when the policy specifies TERMTYPE=ELEMTERM and the IXCARM macro specifies ELEMBIND=CURSYS.

ALLTERM
Automatic restart manager is to restart this element for all unexpected failures as appropriate. The ELEMBIND keyword determines which types of failures are appropriate to restart the element.

ELEMTERM
Automatic restart manager is to restart the element when the element fails but not when the system on which the element is registered is removed from the sysplex or unexpectedly terminates. TERMTYPE=ELEMTERM cannot be specified with ELEMBIND=CURSYS.

SYSTERM
Automatic restart manager is to restart this element when the system on which the element is registered is removed from the sysplex or unexpectedly terminates, but not when the element unexpectedly terminates.

Note: The installation can override this parameter by specifying TERMTYPE for this element in an installation-written policy.

ABEND Codes

None.

Return and Reason Codes

When control returns from IXCARM:
- GPR 15 (and *retcode*, if you coded the RETCODE parameter) contains a return code.
- GPR 0 (and *rsncode*, if you coded the RSNCODE parameter) contains a reason code, if GPR 15 contains a non-zero return code.

The IXCYARM macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCARMRC0
4	IXCARMRC4
8	IXCARMRC8
C	IXCARMRC12
10	IXCARMRC16

Table 3. Return and Reason Codes for the IXCARM Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Equate Symbol:** None. **Meaning:** IXCARM completed successfully. If this was a REGISTER request, this was the first time the element registered with automatic restart management. On a DEREGISTER request, the request was either performed successfully, or the caller attempted to deregister an element that was either not valid or no longer registered. **Note:** This return code could also be received if a timeout occurred during restart processing. Automatic restart management considers this the initial registration of this element. **Action:** None. However, if this element had been deregistered due to an error during restart processing, some cleanup may be necessary.
04	104	**Equate Symbol:** IXCARMPERJCL **Meaning:** IXCARM REQUEST=REGISTER was issued by an element that is being restarted with the same JCL or command text that was used for the previous start of this job. **Action:** None; however, the element might need to perform some cleanup to continue processing.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
04	108	**Equate Symbol:** IXCARMNEWJCL **Meaning:** IXCARM REQUEST=REGISTER was issued by an automatic restart management element that is being restarted with JCL, command text that was provided by an automatic restart management exit or by the automatic restart manager policy, or start text that was specified on the original IXCARM REQUEST=REGISTER. **Action:** None; however, the element might need to perform some cleanup to continue processing.
04	204	**Equate Symbol:** IXCARMPREDTIMEOUT **Meaning:** IXCARM REQUEST=WAITPRED was issued but the predecessor element did not issue an IXCARM REQUEST=READY within its specified time interval. **Action:** None required. However, if your program cannot run without the predecessor element, then some installation-defined action may be necessary.
04	304	**Equate Symbol:** IXCARMREADYTIMEOUT **Meaning:** The IXCARM REQUEST=READY completed but a predecessor of this element did not issue an IXCARM REQUEST=READY within its specified time interval. This reason code is issued only if the caller did not issue IXCARM REQUEST=WAITPRED previously, but has predecessor elements defined in the automatic restart manager policy. **Action:** None required. However, if your program cannot run without the predecessor element, then some installation-defined action may be necessary.
08	14	**Equate Symbol:** IXCARMNOTREG **Meaning:** Program error. Either the caller is not a registered element of the automatic restart manager or the element represents an abstract resource (the element registered with ELEMBIND=CURSYS) and did not specify a valid RMTOKEN on an IXCARM READY, WAITPRED, ASSOCIATE, or DEREGISTER request. Note that for elements that represent jobs or started tasks (the element registered with ELEMBIND=CURJOB), specifying RMTOKEN on a READY, WAITPRED, and ASSOCIATE request is not allowed and results in this reason code. Specifying RMTOKEN on a DEREGISTER request is not allowed and results in this reason code unless the request is issued from the master address space. **Action:** A request other than IXCARM REQUEST=REGISTER was issued out of sequence. Ensure that your program issues an IXCARM REQUEST=REGISTER prior to requesting any other functions from the automatic restart manager.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	18	**Equate Symbol:** IXCARMINVANSADDR **Meaning:** Program error. The system cannot access the answer area provided with this request. The element was successfully registered, but the answer data could not be returned. **Action:** Ensure that: • The answer area address has not been overlaid. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the answer area address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	1C	**Equate Symbol:** IXCARMINVANSALET **Meaning:** Program error. The ALET that qualifies the address of the answer area is not associated with a valid entry on the DU-AL. **Action:** Ensure that: • The SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the answer area address was specified using explicit register notation, the corresponding access register was updated accordingly. • This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	20	**Equate Symbol:** IXCARMINVRMTADDR **Meaning:** Program or environmental error. The system cannot access the RMTOKEN value. **Action:** Ensure that • The token area has not been overlaid. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the token address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	24	**Equate Symbol:** IXCARMINVRTMALET **Meaning:** Program or environmental error. The ALET that qualifies the address of the RMTOKEN is not associated with a valid entry on the DU-AL. **Action:** Ensure that: • The SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the answer area address was specified using explicit register notation, the corresponding access register was updated accordingly. • This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	2C	**Equate Symbol:** IXCARMINVELEMNAME **Meaning:** Program error. For IXCARM REQUEST=REGISTER, the name specified for ELEMENT was not valid. For IXCARM REQUEST=ASSOCIATE, the name specified for TELEMENT was not valid. **Action:** Verify that the element name was not inadvertently overlaid. The element name must adhere to the rules listed under the description of the ELEMENT parameter. For REQUEST=ASSOCIATE, TELEMENT must be the name of an element that is already registered with automatic restart management.
08	30	**Equate Symbol:** IXCARMREQUESTOVERLAP **Meaning:** Program error. An IXCARM request for an element registered with or defaulted to ELEMBIND=CURJOB is already being processed for this address space. The IXCARM request cannot be issued until a previous IXCARM request from this address space has been completed. **Action:** An IXCARM request cannot be issued until all previous IXCARM requests from this address space have completed. Check your protocol to determine how two requests could be outstanding at the same time.
08	34	**Equate Symbol:** IXCARMAMODE24 **Meaning:** Program error. The IXCARM macro was issued in 24-bit addressing mode. **Action:** IXCARM must be invoked in 31-bit addressing mode. Correct the problem and rerun the program.
08	40	**Equate Symbol:** IXCARMRSVNOT0 **Meaning:** Program error. A reserved field in the parameter list is not zero. **Action:** Verify that: • The parameter list was not inadvertently overlaid. • The parameter list was initialized. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • The PLISTVER specified is correct.
08	A0	**Equate Symbol:** IXCARMINVR0 **Meaning:** System error. **Action:** Make sure your program was assembled with the correct macro library for the release of MVS your program is running on. If the macro version is correct, retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	A4	**Equate Symbol:** IXCARMR0TYPECONFL **Meaning:** System error. **Action:** Verify that: • The parameter list was not inadvertently overlaid. • The parameter list was initialized. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. If everything is verified, retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
08	100	**Equate Symbol:** IXCARMINVPLISTALET **Meaning:** Program error. The ALET that qualifies the address of the parameter list is not associated with a valid DU-AL entry. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • The SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. • The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	104	**Equate Symbol:** IXCARMBADVERSION **Meaning:** Program error. The version number specified in the IXCARM parameter list is not valid for the release of MVS this program is running on. **Action:** Ensure that your program: • Specified the correct version on the PLISTVER parameter. • Did not overlay the parameter list storage. • Was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCARMBADREQUEST **Meaning:** Program error. The function specified on the REQUEST parameter of the IXCARM macro is not valid. **Action:** Verify that: • The parameter list was not inadvertently overlaid. • Your program was assembled with the correct macro library for the release of MVS your program is running on.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	10C	**Equate Symbol:** IXCARMPARMERR **Meaning:** Program error. An error occurred when the system tried to access the parameter list. **Action:** Take one or more of the following actions: • Ensure that the parameter list address has not been overlaid. • Ensure that the correct parameter list storage area was specified. • If your program is running in AR ASC mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCARM macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area must be either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Ensure that the parameter list storage area was not inadvertently freed by your program. • Ensure that your program was assembled with the correct macro library for the release of MVS on which your program is running.
08	110	**Equate Symbol:** IXCARMSTARTERR **Meaning:** Program error. An error occurred when the system tried to access the start text. **Action:** Ensure that: • The start text address has not been overlaid. • The correct STARTTXT address was specified. • The STARTTXT storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the start text address was specified using explicit register notation, the corresponding access register was updated accordingly. – The start text is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	114	**Equate Symbol:** IXCARMSTARTLEN **Meaning:** Program error. The length of the start text is not valid. **Action:** Ensure that the parameter list was not inadvertently overlaid. Valid lengths for the start text are from 0 through 126.
08	118	**Equate Symbol:** IXCARMNOTTASKMODE **Meaning:** Program error. The caller is not running in task mode. **Action:** Correct your program so that it issues IXCARM only while in task mode.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	11C	**Equate Symbol:** IXCARMNOTENABLED **Meaning:** The caller is not enabled. **Action:** Correct your program so that it does not issue IXCARM while it is disabled.
08	120	**Equate Symbol:** IXCARMHASLOCK **Meaning:** Program error. The caller of IXCARM holds a lock. **Action:** Correct your program so that it does not issue IXCARM while it is holding a lock.
08	124	**Equate Symbol:** IXCARMHASEUTFRR **Meaning:** Program error. The caller of IXCARM has an EUT FRR established. **Action:** Correct your program so that it does not issue IXCARM while it has an EUT FRR established. An ESTAE should be used if recovery needs to be established for this program.
08	128	**Equate Symbol:** IXCARMJESERR **Meaning:** Program error. A REGISTER request failed because of an error in JES. **Action:** Analyze problem with JES.
08	12C	**Equate Symbol:** IXCARMJOURNAL **Meaning:** Program error. The caller is a candidate for checkpoint/restart and is not eligible to be restarted by automatic restart management. **Action:** If you want to use automatic restart management to restart your jobs, you cannot use a checkpoint/restart.
08	130	**Equate Symbol:** IXCARMINVELEMTYPE **Meaning:** Program error. The name specified for the element type was not valid. **Action:** Make sure the element type specified adheres to the rules outlined under the description of the ELEMTYPE parameter.
08	134	**Equate Symbol:** IXCARMWRONGCALLERTYPE **Meaning:** Program error. The caller was neither a started task nor a batch job, but specified a bind to the current batch job or started task. **Action:** Register the element with a bind to the current system.
08	138	**Equate Symbol:** IXCARMCANCELLED **Meaning:** Environmental error. A register request was received from a program that is in an address space that is being cancelled. The CANCEL command was issued without the ARMRESTART parameter, therefore, the registration of this element is not allowed. **Action:** None; however, you should make sure that your program is not issuing the IXCARM macro with the REQUEST=REGISTER parameter from a recovery routine or resource manager.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	13C	**Equate Symbol:** IXCARMRACRFAIL **Meaning:** Environmental error. The issuer of the IXCARM macro does not have the required SAF authorization. **Action:** Determine why the program issuing the IXCARM macro does not have the proper SAF authorization. If this program was restarted with new command text (specified in the policy, on the STARTTXT parameter of the IXCARM macro, or in the element-restart installation exit), verify that the command text did not cause this problem.
08	140	**Equate Symbol:** IXCARMINVTERMTYPE **Meaning:** Program error. The TERMTYPE value specified is not valid. **Action:** Ensure that: • The parameter list address has not been overlaid. • If your program is running in AR ASC mode: – Your program specified SYSSTATE ASCENV=AR before issuing the IXCARM macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • The parameter list storage area was not inadvertently freed by your program. • Your program was assembled with the correct macro library for the release of MVS your program is running on.
08	144	**Equate Symbol:** IXCARMINVRESTTIMEOUT **Meaning:** Program error. The RESTARTTIMEOUT value specified is not valid. **Action:** Ensure that: • The parameter list address has not been overlaid. • If your program is running in AR ASC mode: – Your program specified SYSSTATE ASCENV=AR before issuing the IXCARM macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • The parameter list storage area was not inadvertently freed by your program. • Your program was assembled with the correct macro library for the release of MVS your program is running on.
08	148	**Equate Symbol:** IXCARMSAVEFAIL **Meaning:** Environmental error. Register request prohibited by JES. **Action:** This element cannot be restarted by automatic restart management.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	14C	**Equate Symbol:** IXCARMBATCHSTARTTXT **Meaning:** Program error. An element registering under a batch job with a bind to the job specified STARTTXT with its register request. **Action:** Do one of the following: • If you have a batch job that needs different JCL if it is restarted, then use the policy to point to this information. • Code an automatic restart manager element restart exit that will point to the different JCL if it is restarted.
08	150	**Equate Symbol:** IXCARMELEMNAMEINUSE **Meaning:** Program error. The element name specified for this REGISTER request is already registered. **Action:** Element names must be unique across a sysplex. To determine what element names are already in use, you could issue the IXCQUERY macro and request ARMSTATUS. This will provide you with a list of element names that are already in use.
08	154	**Equate Symbol:** IXCARMADDRSPACEDUP **Meaning:** Program error. An element with a bind to the batch job or started task is already registered with the automatic restart manager. Only one element per batch job or started task can register with a bind specification of CURJOB. **Action:**Ensure that the application is not registering more than once or use the ELEMBIND=CURSYS keyword option if appropriate.
08	158	**Equate Symbol:** IXCARMEXITPARM **Meaning:** Program error or environmental error. The system cannot access the event exit parameter list specified in the EVENTEXITPL parameter. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The event exit parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	15C	**Equate Symbol:** IXCARMEXITLEN **Meaning:** Program error. The value specified in EXITPLLEN has exceeded the maximum length. **Action:** See the description of the EXITPLLEN parameter for the limits on this length. Ensure that: • The parameter list length has not been overlaid. • The correct parameter list length was specified. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The event exit parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	160	**Equate Symbol:** IXCARMEXITNAME **Meaning:** Program error or environmental error. The system cannot access the name of the event exit. **Action:** Ensure that: • The correct event exit name was specified. • The event exit is available on every system in the sysplex that could be the target of a restart by automatic restart management. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The event exit parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL.
08	164	**Equate Symbol:** IXCARMINVEVENTEXIT **Meaning:** Program error. The name specified for the event exit routine is not a valid MVS load module name. **Action:** Make sure the event exit name specified adheres to the rules listed under the EVENTEXIT parameter description.
08	168	**Equate Symbol:** IXCARMINVASYNCREQ **Meaning:** Program error. An IXCARM request requiring asynchronous processing is invalid in the issuer's address space. **Action:** Check your program to ensure that you are issuing the IXCARM request in a valid environment. For example, do not issue IXCARM in an installation restart exit.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	16C	**Equate Symbol:** IXCARMINVELEMBIND **Meaning:** Program error. The ELEMBIND value specified is not valid, or ELEMBIND=CURSYS was specified with TERMTYPE=ELEMTERM. **Action:** Check your program to ensure that it specified a valid value for ELEMBIND and did not specify ELEMBIND=CURSYS with TERMTYPE=ELEMTERM.
08	1A8	**Equate Symbol:** IXCARMRSVREGFDS **Meaning:** Program error. A REGISTER request contained fields that did not apply to REGISTER and were not zero. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Your program was assembled with the correct macro library for the release of z/OS on which your program is running. Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel. Also supply IBM with the IXCARM macro invocation and/or generated parameter list to diagnose the problem.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	204	**Equate Symbol:** IXCARMBADWAITPRED **Meaning:** Program or environmental error. An IXCARM REQUEST=WAITPRED was received from an element that is already in an Available or Available-To state. An element is placed in an Available state when the element has successfully completed an IXCARM REQUEST=READY request. An element is placed in an Available-To state when the element was restarted by ARM, reregistered, but did not complete an IXCARM REQUEST=READY within the Automatic Restart Manager policy-defined READY_TIMEOUT value. • This is a programming error if the element had already successfully completed an IXCARM REQUEST=READY request. • This is an environmental error if the ARM policy did not allot enough time for the element to complete its initialization and an IXCARM REQUEST=READY request. **Action:** Continue processing. ARM already considers the element to be in an available state. As such, any predecessor elements would be available as well. Note that elements that are at a higher level than your element might think your services are available prematurely. Check your program to determine if it had already issued a READY request. Use the IXCQUERY service to determine the current state of the element (AVAILABLE or AVAILABLE-TO). You can also use the DISPLAY XCF,ARM command for manual debugging. If the element has truly timed out (AVAILABLE-TO), consider changing your logic to continue processing.
08	2A8	**Equate Symbol:** IXCARMRSVWTPFDS **Meaning:** Program error. An IXCARM WAITPRED request contained fields not applying to WAITPRED that were not zero. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Your program was assembled with the correct macro library for the release of OS/390 on which your program is running. Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel. Also supply IBM with the IXCARM macro invocation and/or generated parameter list to diagnose the problem.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	304	**Equate Symbol:** IXCARMBADREADY **Meaning:** Program error. An IXCARM REQUEST=READY was received from an element that had already issued an IXCARM REQUEST=READY. **Action:** Check your program to determine why the READY request was issued more than once.
08	3A8	**Equate Symbol:** IXCARMRSVRDYFDS **Meaning:** System error. An IXCARM READY request contained required fields that were not zero. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Your program was assembled with the correct macro library for the release of OS/390 on which your program is running. Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel. IBM will also need the IXCARM macro invocation and/or generated parameter list to diagnose the problem.
08	404	**Equate Symbol:** IXCARMDUPASSOC1 **Meaning:** Program error. An IXCARM REQUEST=ASSOCIATE was issued by an element that was already associated with an element. **Action:** An element may be associated only with one other element. Check your program to determine why the ASSOCIATE request was issued more than once from this element.
08	408	**Equate Symbol:** IXCARMBADTARGETELEM **Meaning:** Program error. The element specified by the TELEMENT parameter on an IXCARM REQUEST=ASSOCIATE was not a registered element. **Action:** The TELEMENT specified in an ASSOCIATE request must be an element that is already registered with the automatic restart manager. Make sure the element name is correct. You can issue the IXCQUERY macro with the ARMSTATUS parameter for a list of all the elements that have registered with the automatic restart manager.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	40C	**Equate Symbol:** IXCARMDUPASSOC2 **Meaning:** Program error. The element specified by the TELEMENT parameter on an IXCARM REQUEST=ASSOCIATE was already associated with another element. **Action:** An element may be associated only with one other element. Check your program to determine why the ASSOCIATE request was issued more than once for this element.
08	414	**Equate Symbol:** IXCARMSELFASSOC **Meaning:** Program error. The element specified by the TELEMENT parameter in an IXCARM REQUEST=ASSOCIATE is the same element that issued the ASSOCIATE request. **Action:** An element cannot be associated with itself. Correct the TELEMENT value and reissue the request.
08	4A8	**Equate Symbol:** IXCARMRSVASSFDS **Meaning:** System error. An IXCARM ASSOCIATE request contained required fields that were not zero. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Your program was assembled with the correct macro library for the release of OS/390 on which your program is running. Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel. IBM will also need the IXCARM invocation and/or generated parameter list to diagnose the problem.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	5A8	**Equate Symbol:** IXCARMRSVDRGFDS **Meaning:** System error. An IXCARM DEREGISTER request contained required fields that were not zero. **Action:** Ensure that: • The parameter list address has not been overlaid. • The correct parameter list address was specified. • The parameter list storage area was not inadvertently freed by your program. • If IXCARM was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. • Your program was assembled with the correct macro library for the release of OS/390 on which your program is running. Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel. IBM will also need the IXCARM invocation and/or generated parameter list to diagnose the problem.
08	5B0	**Equate Symbol:** IXCARMWRONGELEMONREREG **Meaning:** The element that has attempted to register has done so in an address space that was created for the restart of another element. Only the restarted element can re-register in the current address space. ARM rejects the registration because as part of restarting the related job or started task address space, ARM propagates resources related to the member from the previous instance of the member. Currently the system symbolic substitution table is the only resource that is propagated. Allowing the element to register with ARM might cause an incorrect resource to be used by the element. **Action:** Ensure that: • You register with ARM as soon as possible so the condition can be identified. • You use the same ELEMENT name when the application is restarted anywhere in the sysplex. For example, using a system name as part of the element name would not work because the element name then would not match when the element was restarted on another system. If the element name was not correct, re-attempt the IXCARM registration with the correct name. • You terminate your application and inform the operator that your application element was incorrectly started or restarted by a method other than ARM.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	5B4	**Equate Symbol:** IXCARMWRONGADDRONREREG **Meaning:** The element that has attempted to re-register has done so in an address space other than the one that was created for the re-registering element. The element can only re-register in the address space in which the override restart start text was issued. ARM rejects the registration because as part of restarting the related job or started task address space, ARM propagates resources related to the member from the previous instance of the member. Currently the system symbolic substitution table is the only resource that is propagated. Allowing the element to register with ARM might cause an incorrect resource to be used by the element. **Action:** Ensure that: • You register with ARM as soon as possible so the condition can be identified. • You use the same ELEMENT name when the application is restarted anywhere in the sysplex. For example, using a system name as part of the element name would not work because the element name then would not match when the element was restarted on another system. If the element name was not correct, re-attempt the IXCARM registration with the correct name. • You terminate your application and inform the operator that your application element was incorrectly started or restarted by a method other than ARM.
08	5B8	**Equate Symbol:** IXCARMREREGAFTERTIMOUT **Meaning:** The element that has attempted to register has done so in an address space that was created for the restart of another element. However, the element for which the address space was initially created is no longer known to ARM. This is probably due to the restart of the element having timed out. ARM rejects the registration because as part of restarting the related job or started task address space, ARM propagates resources related to the member from the previous instance of the member. Currently the system symbolic substitution table is the only resource that is propagated. Allowing the element to register with ARM might cause an incorrect resource to be used by the element. **Action:** Ensure that: • You register with ARM as soon as possible so the condition can be identified. • You use the same ELEMENT name when the application is restarted anywhere in the sysplex. For example, using a system name as part of the element name would not work because the element name then would not match when the element was restarted on another system. If the element name was not correct, re-attempt the IXCARM registration with the correct name. • You terminate your application and inform the operator that your application element was incorrectly started or restarted by a method other than ARM.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	5BC	**Equate Symbol:** IXCARMUNAUTHEVENTEXIT **Meaning:** Users who are both in problem state and problem key cannot specify an event exit keyword on registration. **Action:** Remove the EVENTEXIT parameter from the IXCARM invocation and resubmit.
08	5C0	**Equate Symbol:** IXCARMUNAUTHSTARTTXT **Meaning:** Users who are both in problem state and problem key cannot specify a start text keyword on registration. **Action:** Remove the STARTTXT parameter from the IXCARM invocation and resubmit.
0C	5C4	**Equate Symbol:** IXCARMUNAUTHRMTOKEN **Meaning:** Program error. Users who are both in problem state and problem key cannot specify the RMTOKEN keyword on any request. **Action:** Remove the RMTOKEN keyword, or switch to supervisor state or system key before using it.
0C	04	**Equate Symbol:** IXCARMNOARM **Meaning:** Environmental error. The IXCARM macro was issued on a system that does not support the automatic restart manager. **Action:** Issue the IXCARM macro only on systems that have MVS SP5.2.0 (or higher) and either JES2 SP5.2.0 (or higher), or JES3 SP5.2.1 (or higher), and have access to an ARM couple data set. Consult your system programmer to determine the level of the system you are running on and which systems have connectivity to the ARM couple data set.
0C	0C	**Equate Symbol:** IXCARMNOESTAE **Meaning:** System error. The request is not processed. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
0C	C0	**Equate Symbol:** IXCARMFDSERR1 **Meaning:** Environmental error. Internal error while trying to access the automatic restart manager's couple data set. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
0C	C4	**Equate Symbol:** IXCARMFDSERR2 **Meaning:** System error. Internal error while trying to access the automatic restart manager's couple data set. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	C8	**Equate Symbol:** IXCARMFDSERR3 **Meaning:** System error. Internal error while trying to access the automatic restart manager's couple data set. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
0C	CC	**Equate Symbol:** IXCARMBADTESTART **Meaning:** System error. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
0C	104	**Equate Symbol:** IXCARMMAXUSERS **Meaning:** Environmental error. The maximum number of automatic restart management users has been reached. **Action:** Retry the request at least once. If the problem persists, consult your system programmer to determine if the maximum number of elements defined in the automatic restart management couple data set should be increased or if the limit on the number of elements has been reached. Your application should cover cases in which the automatic restart management couple data set has no available room. Before running or installing your application, you should make the system programmer aware of your element's requirements.
0C	160	**Equate Symbol:** IXCARMNOCDS **Meaning:** Environmental error. The system on which the IXCARM macro was issued does not have access to an automatic restart managememt couple data set. **Action:** Contact the system programmer to determine which systems have a couple data set for the automatic restart manager.
0C	164	**Equate Symbol:** IXCARMBADJOB **Meaning:** Environmental error. JES could not support automatic restart manager requests for this job. A unit of work other than a normal batch job or started task has attempted to register with automatic restart manager without specifying ELEMBIND=CURSYS. The registration was rejected. **Action:** Register the element with a bind to the current system.
0C	168	**Equate Symbol:** IXCARMSAFNOTDEFINED **Meaning:** Environmental error. Problem state and problem key users cannot use IXCARM without having a security profile. **Action:** Ensure that the proper IXCARM.elemtype.elemname resource profile for the unauthorized application is defined to RACF or another security product.

Table 3. Return and Reason Codes for the IXCARM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	16C	**Equate Symbol:** IXCARMNOSAFAUTH **Meaning:** Environmental error. The installed security product indicated that the user does not have authorized access to the IXCARM facility or the secure entity. The entity is made up of the element type and element name. **Action:** Ensure that the unauthorized application has UPDATE access to the IXCARM.elemtype.elemname resource profile in the FACILITY class.
10	04	**Equate Symbol:** IXCARMARMERR **Meaning:** System error. The element was deregistered due to an internal error. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
10	08	**Equate Symbol:** IXCARMUNKERR **Meaning:** System error. An unexpected error occurred. Automatic restart management will attempt to deregister this element. **Action:** Retry the request at least once. If it fails again, try to deregister the element or issue the IXCQUERY macro to determine the state of this element. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
10	A0	**Equate Symbol:** IXCARMPCCERROR **Meaning:** System error. An unexpected error occurred. Automatic restart management will attempt to deregister this element. **Action:** Retry the request at least once. If it fails again, try to deregister the element or issue the IXCQUERY macro to determine the state of this element. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.

Example

See *z/OS MVS Programming: Sysplex Services Guide* in the automatic restart management chapter for an example of this macro.

IXCCREAT — Define a Member to XCF

Description

The IXCCREAT macro defines a member to the cross-system coupling facility (XCF). Use it to define the group name and member name, and optionally, place a value in the member's user state field. IXCCREAT also:

- Places the member in the created state.
- Assigns a member token to the new member.
- Activates permanent status recording for the member.
- If active members have a group user-routine, notifies those members about the existence of the created member.

For more information on XCF see *z/OS MVS Programming: Sysplex Services Guide*.

Before the member can use the signalling or monitoring services of XCF, it must issue the IXCJOIN macro with the LASTING=YES parameter. Other active members in the specified group can issue the IXCQUERY macro to access the name, status, and user state field of the created member. They can issue the IXCSETUS macro to change the user state of a created member.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0-7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCCREAT. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

The IXCYQUAA mapping macro provides the format of the area that the ANSAREA parameter points to. If you intend to use that area, include that mapping macro in your program.

Restrictions

The caller can have no enabled, unlocked task (EUT) FRRs established.

Do not issue the IXCCREAT macro on a system that is in XCF-local mode. In XCF-local mode, XCF does not support permanent status recording.

Input Register Information

Before issuing the IXCCREAT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Syntax Diagram

The syntax of the IXCCREAT macro is as follows:

IXCCREAT diagram

```
►►──IXCCREAT──b──GRPNAME=grpname──,ANSAREA=ansarea──,ANSLEN=anslen──,MEMNAME=memname──────────────►

   ┌─,USTATE=NO_USTATE──────────────┐
►──┼────────────────────────────────┼──┬─────────────────────┬──┬─────────────────────┬───────────►
   └─,USTATE=ustate───,USLEN=uslen──┘  └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘

   ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────────────────┼──────────────────────────────────────────────────────►◄
   │                    ┌─,0D──────┐       │
   ├─,MF=(L─,mfctrl─────┼──────────┼─)─────┤
   │                    └─,mfattr──┘       │
   │                    ┌─,COMPLETE─┐      │
   └─,MF=(E─,mfctrl─────┼───────────┼─)────┘
                        └─,COMPLETE─┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ANSAREA=*ansarea*

Use this output parameter to specify a storage area to contain information from the request. Member information includes the group name, member name, member token, and any data you place in the user state field specified on USTATE. You will use the member token as input to other XCF macros.

The IXCYQUAA mapping macro provides the format of the member information area. The member information area can be in the primary address space or be addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

Use the ANSLEN parameter to specify the length of the area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the member information will go.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the answer area specified by ANSAREA. The size of the area must be greater than or equal to the length of the data area for a single member record returned by IXCQUERY plus the length of the user state field. The field QUAMLEN in the IXCYQUAA mapping macro contains this value (the length of the member record plus the length of the user state field).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the member information area (ANSAREA).

GRPNAME=*grpname*

Use this input parameter to specify the name of the group to which the member is to belong. The group name must be eight characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, and national characters ($, # and @). To avoid using the names IBM uses for its XCF groups, do not begin group names with the letters A through I or the character string **SYS**. Also, do not use the name UNDESIG, which is reserved for use by the system programmer in your installation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the name of the XCF group.

,MEMNAME=*memname*

Use this input parameter to specify the name you chose for the member. The name must be 16 characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, and national characters ($, # and @).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the XCF member name.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,USLEN=*uslen*
Use this input parameter to specify the length in bytes of the user state data that you provide on the USTATE parameter. The length must be from 1 to 32 bytes. If you specify less than 32 bytes, XCF pads the remainder of the user state field, up to 32 bytes, on the right with zeros. IXCQUERY always returns the full 32 bytes, and group user-routines always receive the full 32 bytes. If you code USTATE, USLEN is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the user state data (USTATE).

,USTATE=NO_USTATE
,USTATE=*ustate*
Use this input parameter to specify the data that you want XCF to place in the user state field associated with the member. Use the USLEN parameter to specify the length of the user state data.

If you do not specify USTATE, or if you specify USTATE=NO_USTATE, XCF sets the user state field to zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) that contains the user state information.

ABEND Codes

None.

Return and Reason Codes

When the IXCCREAT macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING

8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 4. Return and Reason Codes for the IXCCREAT Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
00	None.	**Meaning:** IXCCREAT completed successfully; XCF places member information in the ANSAREA. **Action:** None.
04	00	**Equate Symbol:** IXCCREATRSNFIRSTMEMBER **Meaning:** This is the first member XCF is defining in the group; XCF places member information in the ANSAREA. **Action:** None.
08	04	**Equate Symbol:** IXCCREATRSNALREADYCREATED **Meaning:** Program error. The member already exists in a created state. **Action:** None required. However, if your program did not expect this return code, you might want to do one of the following: • Choose a different unique name for your new member. • Use the IXCDELET service to delete the current instance of the member, and then reissue the IXCCREAT service to create a new instance of the member. XCF will assign a unique MEMTOKEN to the new instance. • Use the IXCSETUS service to update the member's user state.
08	08	**Equate Symbol:** IXCCREATRSNISACTIVE **Meaning:** Program error. The member already exists in an active state. **Action:** None required. However, if you did not expect this return code, you might want to do one of the following: • Choose a different unique name for your new member. • Use the IXCLEAVE service to remove the current instance of the member, and then reissue the IXCCREAT service to create a new instance of the member. XCF will assign a new unique MEMTOKEN to the new instance.
08	0C	**Equate Symbol:** IXCCREATRSNISQUIESCED **Meaning:** Program error. The member already exists in a quiesced state. **Action:** None required. However, if you did not expect this return code, you might want to do one of the following: • Choose a different unique name for your new member. • Use the IXCDELET service to delete the current instance of the member, and then reissue the IXCCREAT service to create a new instance of the member. XCF will assign a new unique MEMTOKEN to the new instance.

Table 4. Return and Reason Codes for the IXCCREAT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	10	**Equate Symbol:** IXCCREATRSNISFAILED **Meaning:** Program error. The member already exists in a failed state. **Action:** None required. However, if you did not expect this return code, you might want to do one of the following: • Choose a different unique name for your new member. • Use the IXCDELET service to delete the current instance of the member, and then issue the IXCCREAT service again to create a new instance of the member. XCF will assign a new unique MEMTOKEN to the new instance.
08	14	**Equate Symbol:** IXCCREATRSNGRPNAMEBAD **Meaning:** Program error. The group name is not valid. **Action:** Correct the group name, and retry the request.
08	18	**Equate Symbol:** IXCCREATRSNMEMNAMEBAD **Meaning:** Program error. The member name is not valid. **Action:** Correct the member name, and retry the request.
08	3C	**Equate Symbol:** IXCCREATRSNANSAREAINCOMPLETE **Meaning:** Program error. The caller specified ANSAREA or ANSLEN incorrectly. Even though the member might have been placed in a created state, the ANSAREA was nonaddressable or the ANSLEN was not large enough. **Action:** Check the two high-order bytes of this reason code fullword *xxyy*003C for the return code *xx* (either 00 or 04) and reason code *yy* the caller would have received if the caller had coded those parameters correctly. Take action as described by the corresponding return and reason code. You might want to abnormally end your program or take some other action that will record the problem. You should correct your program to ensure that the ANSAREA is addressable and that the ANSLEN is large enough. See the description of these keywords for their requirements. If the return and reason code indicate that the member was placed in a created state, you can use the IXCQUERY service to get the information that would have been returned in the ANSAREA. **Note:** You should specify a member name or a unique user state value on the IXCCREAT invocation. If you request to have XCF define the member name, you might not be able to use the IXCQUERY macro to determine which member was created on your behalf.
08	40	**Equate Symbol:** IXCCREATRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 4. Return and Reason Codes for the IXCCREAT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	100	**Equate Symbol:** IXCCREATRSNPLISTBADALET **Meaning:** Program error. Your program is not running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • Your program was not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR prior to issuing the IXCCREAT macro. • The ALET for the parameter list is a valid public entry on the DU-AL or zero (primary address space ALET).
08	104	**Equate Symbol:** IXCCREATRSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCCREATRSNPLISTBADFUNCTION **Meaning:** Program error or environmental error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	10C	**Equate Symbol:** IXCCREATRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCCREAT macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.

Table 4. Return and Reason Codes for the IXCCREAT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	110	**Equate Symbol:** IXCCREATRSNUSTATEBADSTG **Meaning:** Program error. XCF could not access the USTATE value. **Action:** Take one or more of the following actions: • Ensure that the correct USTATE address was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCCREAT macro. – Ensure that the USTATE ALET is correct. • Ensure that the USTATE storage area was not inadvertently freed by your program.
08	114	**Equate Symbol:** IXCCREATRSNUSLENBADVALUE **Meaning:** Program error. The USLEN value is less than 1 or greater than 32. **Action:** Correct the USLEN, and retry the request.
08	118	**Equate Symbol:** IXCCREATRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCCREAT only while in task mode.
08	11C	**Equate Symbol:** IXCCREATRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCCREAT only while enabled.
0C	04	**Equate Symbol:** IXCCREATRSNMAXGROUPS **Meaning:** Environmental error. Your request would have created the first member in a new XCF group. The new group could not be created because the maximum number of groups as defined by the current couple data set already exists. **Action:** Retry the request at least once. If the problem persists, consult your system programmer to determine if the number of groups defined in the couple data set should be increased, or if the XCF limit on the number of groups has been reached. Your application should cover cases in which the couple data set has no available room. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements.

Table 4. Return and Reason Codes for the IXCCREAT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0C	08	**Equate Symbol:** IXCCREATRSNMAXMEMBERS **Meaning:** Environmental error. The maximum number of members in the group already exists. The maximum number of members per group is defined by the installation when the couple data set is formatted. **Action:** Retry the request at least once. If the problem persists, consult your system programmer to determine if the maximum number of members defined in the couple data set should be increased, or if the XCF limit on the number of members has been reached. Your application should cover cases in which no room is available within the group. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements.
0C	10	**Equate Symbol:** IXCCREATRSNPARTITIONING **Meaning:** Environmental error. The system is being removed from the sysplex, and all IXCCREAT requests are permanently suspended. **Action:** The action required depends on your application. You might want to prepare your application for system termination. (See “Using the Cross-System Coupling Facility” in *z/OS MVS Programming: Sysplex Services Guide*.)
0C	14	**Equate Symbol:** IXCCREATRSNXCFLOCALMODE **Meaning:** Environmental error. IXCCREAT is not allowed when the system is in XCF-local mode. By default, a member in a created state has permanent status recording. Permanent status recording requires a couple data set in which XCF can retain the member’s status. XCF-local mode does not support a couple data set. **Action:** The action required depends on your application. If XCF-local mode is not supported by your application, you must make the system programmer aware of this requirement. The system programmer will have to ensure that the system is not IPLed in XCF-local mode. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements. See *z/OS MVS Programming: Sysplex Services Guide*.
0C	18	**Equate Symbol:** IXCCREATRSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended; that is, another unit of work attempted to abnormally terminate this task. The state of the IXCCREAT request is unpredictable. **Action:** Determine why this task was being abended.
10	None	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Create a member MEMB2 in a group MYGROUP with a user state of X'11'. MYAREA points to the area where XCF is to return member information. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCCREAT GRPNAME=MYGROUP,ANSAREA=MYAREA,ANSLEN=AREALEN,       X
               MEMNAME=MEMB2,USTATE=STATE,USLEN=LEN,                   X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

RETURN   DS    1F                        RETURN CODE
REASON   DS    1F                        REASON CODE
MYAREA   DS    CL124                     OUTPUT AREA TO CONTAIN DATA    X
                                         RETURNED BY IXCCREAT
MYGROUP  DC    CL8'MYGROUP '             GROUP NAME
MEMB2    DC    CL16'MEMB2           '    MEMBER NAME
STATE    DC    X'11'                     USER STATE VALUE FOR THIS      X
                                         MEMBER
LEN      DC    F'1'                      LENGTH OF USER STATE FIELD
AREALEN  DC    F'124'                    LENGTH OF OUTPUT AREA
```

IXCDELET — Change an XCF Member's State to Not-Defined

Description

The IXCDELET macro allows an authorized routine to change the state of a failed, quiesced, or created cross-system coupling facility (XCF) member to the not-defined state. XCF does not recognize the existence of a member in the not-defined state. Therefore, all future requests against this member will generate a return code indicating that the member no longer exists. For created members, XCF notifies the active members of the group, through their group user-routines, about the change in the member's state.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0-7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or in an address/data space that is addressable through a public entry in the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before IXCDELET. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

The caller can have no enabled, unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCDELET macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

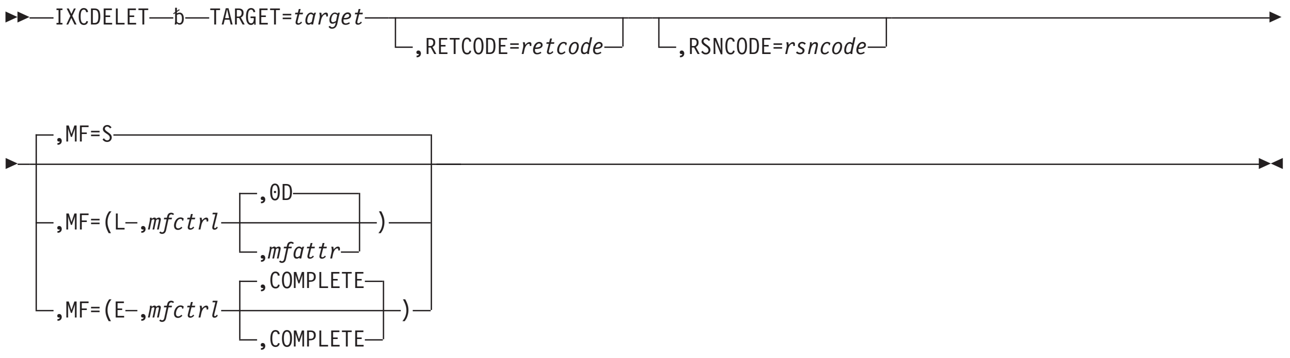
Performance Implications

None.

Syntax Diagram

The syntax of the IXCDELET macro is as follows:

IXCDELET diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

TARGET=*target*

Use this input parameter to specify the 64-bit token of the created, quiesced, or failed member that XCF is to place in the not-defined state. IXCJOIN (or IXCCREAT) provided this token when it activated (or created) the member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the token.

ABEND Codes

None.

Return and Reason Codes

When the IXCDELET macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 5. Return and Reason Codes for the IXCDELET Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** IXCDELET completed successfully; XCF places the member in the not-defined state. **Action:** None.

Table 5. Return and Reason Codes for the IXCDELET Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	04	**Equate Symbol:** IXCDELETRSNNOTDEFINED **Meaning:** Program error. The target member does not exist. **Action:** None required. However, if the target member is expected to exist, make sure TARGET MEMTOKEN is correct, and retry the request.
08	08	**Equate Symbol:** IXCDELETRSNINAPPROPRIATESTATE **Meaning:** Program error. The target member is not in a created, failed, or quiesced state. **Action:** The action required depends on your application. If the member is expected to be in a created, failed, or quiesced state, ensure that the TARGET MEMTOKEN is correct, and retry the request. If the target member can be in another state, use the IXCQUERY service to determine what state the member is in. If the member is not defined to the XCF group, no action is required. If the member is in an active state and should be placed in a not-defined state, you can use the IXCLEAVE or IXCTERM service to place the member in a not-defined state.
08	40	**Equate Symbol:** IXCDELETRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	100	**Equate Symbol:** IXCDELETRSNPLISTBADALET **Meaning:** Program error. Your program is not running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCDELET macro. • The ALET for the parameter list is on the DU-AL or is zero (primary address space ALET).
08	104	**Equate Symbol:** IXCDELETRSNPLISTVERSIONNOTVALID **Meaning:** Program error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.
08	108	**Equate Symbol:** IXCDELETRSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.

Table 5. Return and Reason Codes for the IXCDELET Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	10C	**Equate Symbol:** IXCDELETRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR ASC mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCDELET macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
08	118	**Equate Symbol:** IXCDELETRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCDELET only while in task mode.
08	11C	**Equate Symbol:** IXCDELETRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCDELET only while enabled.
0C	18	**Equate Symbol:** IXCDELETRSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended (i.e.another unit of work attempted to abnormally terminate this task). The state of the IXCDELET request is unpredictable. **Action:** Determine why this task was being abended.
10	None	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Delete a member from a group. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCDELET TARGET=TOKEN3,RETCODE=RETURN,RSNCODE=REASON,MF=S

TOKEN3   DS    CL8                   TOKEN OF MEMBER TO BE PLACED   X
                                     IN NOT-DEFINED STATE
RETURN   DS    1F                    RETURN CODE
REASON   DS    1F                    REASON CODE
```

You can obtain the member token from the QUAMTOKN field in the area returned by IXCCREAT, IXCJOIN, or IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXCJOIN — Place an XCF Member in the Active State

Description

The IXCJOIN macro places a cross-system coupling facility (XCF) member in the active state, associating it with an XCF group. In the active state, the member can use the monitoring and signalling services of XCF. Additionally, IXCJOIN:

- Returns a member token. (Each member, regardless of its previous state, receives a **new** member token.)
- Notifies the active group members that have a group user-routine that the joining member is in an active state.

The member might previously have been in the failed, quiesced, not-defined, or created state.

The required parameters ANSAREA and ANSLEN specify the area where XCF returns member information, including a member token.

With the optional parameters USTATE and USLEN, you can place a value in the user state field. The data in the user state field is provided to the group user-routines.

Optional parameters on IXCJOIN allow you to:
- Associate the joining member with a task, job step task, or address space (MEMASSOC).
- Establish permanent status recording (LASTING=YES).
- Specify the address of the message user-routine (MSGEXIT).
- Specify the address of the group user-routine (GRPEXIT).
- Specify the address of the status user-routine (STATEXIT).
- Specify whether cleanup will be performed when a system leaves the sysplex (SYSCLEANUPMEM).

Additional optional parameters on version 1 of IXCJOIN allow you to:
- Specify whether the member can participate in a message response collection protocol (CANREPLY).
- Specify the address of the message notify user routine (NOTIFYEXIT).
- Specify whether the member supports sending or receiving messages greater than 61K bytes in length (up to 128M bytes) (GT61KMSG).

Status monitoring requires a status field (STATFLD) and an interval value (INTERVAL) that sets the status-checking interval. The interval determines how long the status field can remain unchanged before XCF schedules the status user-routine.

For information about the user-routines, see the chapter on XCF in *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry in the caller’s dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCJOIN. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

The IXCYQUAA mapping macro provides the format of the area that the ANSAREA parameter points to (the QUAMEM section maps the member information). If you intend to use that area, include the IXCYQUAA mapping macro in your program.

Restrictions

The caller can have no enabled, unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCJOIN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if GPR 15 contains a non-zero return code that has applicable reason codes.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Understanding IXCJOIN Version Support

The IXCJOIN macro supports versions 0 and 1.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXCJOIN macro.
- The following keywords are supported by version 1 and subsequent versions of the IXCJOIN macro.

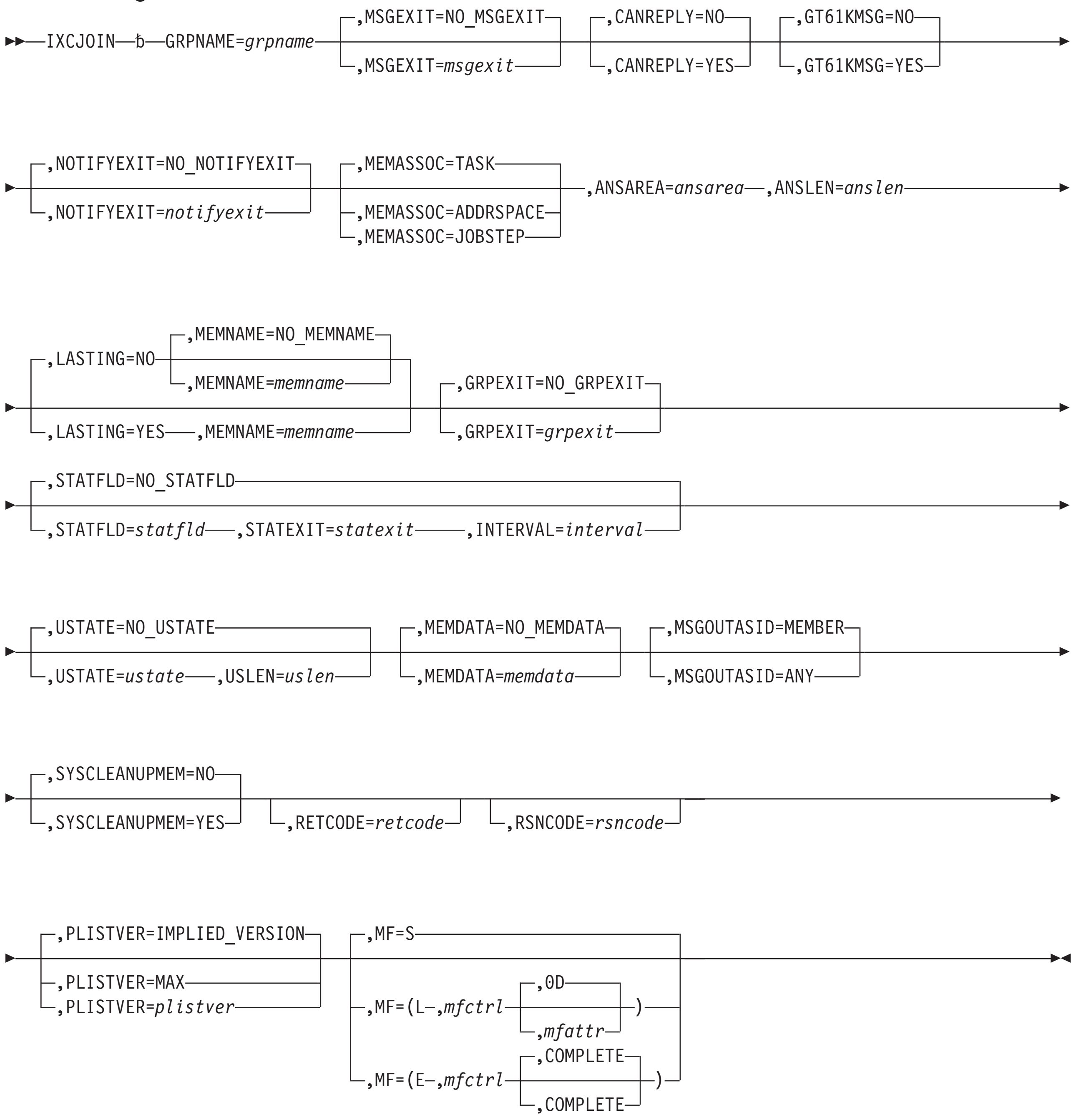
CANREPLY
GT61KMSG
NOTIFYEXIT

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCJOIN macro is as follows:

IXCJOIN diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ANSAREA=*ansarea*

Use this output parameter to specify a storage area to contain member information returned from the request. Member information includes the group name, member name, member token, and any data you place in the user state field specified on USTATE. You will use the member token as input to other XCF macros.

The IXCYQUAA mapping macro, in the QUAMEM section, provides the format of the member information area. The member information area can be in the primary address space or be addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

Use the ANSLEN parameter to specify the length of the area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the system will put the member information.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the answer area specified by ANSAREA. The size of the area must be greater than or equal to the length of the data area for a single member record returned by IXCQUERY plus the length of the user state field. The field QUAMLEN in the IXCYQUAA mapping macro contains this value (the length of the member record plus the length of the user state field).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the member information area (ANSAREA).

,CANREPLY=NO
,CANREPLY=YES

Use this input parameter to specify whether the member can participate in the protocol for XCF-managed response collection for response-required signals.

NO

The member does not participate in XCF-managed response collection. A response-required message will be sent to the member even if it does not participate in the XCF-managed response collection protocol. XCF does not expect the member to recognize or respond to a message that requires an XCF-managed response. XCF response management does not wait for responses from this member.

YES

The member participates in XCF-managed response collection. The member can recognize and respond to a message that requires an XCF-managed response. The member detects when an XCF-managed response is required and sends the response with the IXCMSGO SENDTO=ORIGINATOR service.

The message exit parameter list, mapped by IXCYMEPL, the message notification exit parameter list, mapped by IXCYMNPL, or data, mapped by IXCYMQAA, returned by the Message Control Query Message service (IXCMSGC REQUEST=QUERYMSG) provide the information needed to allow the member to participate in this protocol.

,GRPEXIT=NO_GRPEXIT
,GRPEXIT=*grpexit*

Use this input parameter to specify the address of the optional group user-routine. This routine executes in the primary address space of the issuer of the IXCJOIN macro, in 31-bit addressing mode and in SRB mode. The routine receives control when there is a change in the operational state of other members in the group or systems in the sysplex.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the group user-routine.

GRPNAME=*grpname*

Use this input parameter to specify the name you assign to the group that the member joins. The group name must be eight characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, and national characters ($, # and @). To avoid using the names IBM uses for its XCF groups, do not begin group names with the letters A through I or the character string **SYS**. Also, do not use the name UNDESIG, which is reserved for use by the system programmer in your installation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the name of the XCF group.

,GT61KMSG=NO
,GT61KMSG=YES

Use this input parameter to specify whether the member supports sending or receiving messages greater than 61K bytes in length (62,464 bytes). If so, the member is allowed to use the XCF Message Services for messages up to 128M bytes in length (134,217,728 bytes).

Data returned by the IXCQUERY service indicates whether a member is permitted to send or receive messages greater than 61K bytes.

NO

Messages for this member are limited to a maximum length of 61K bytes. XCF will not allow the member to use the XCF Message-out Service (IXCMSGO) to send messages longer than 61K bytes. Messages sent to this member will be rejected by the XCF Message-out Service if they exceed 61K bytes.

YES

Messages for this member are limited to a maximum length of 128M bytes. XCF will not allow the member to use the XCF Message-out Service (IXCMSGO) to send messages longer than 128M bytes. Messages sent to this member will be rejected by the XCF Message-out Service if they exceed 128M bytes. The member is capable of using the XCF Message-in Service (IXCMSGI) to receive messages of up to a maximum of 128M bytes, although the application might impose a lower message length.

The message exit parameter list, mapped by IXCYMEPL, the message notification exit parameter list, mapped by IXCYMNPL, or data, mapped by IXCYMQAA, returned by the Message Control Query Message service (IXCMSGC REQUEST=QUERYMSG) provide the information needed to allow the member to participate in this protocol.

,INTERVAL=*interval*

Use this input parameter to specify the status-checking interval, in hundredths of seconds, that determines the length of time that can elapse with no change to the status field before scheduling the user status routine. Specify the interval in full seconds; that is, the value must be greater than zero and must be a multiple of 100. If you specify INTERVAL, you must also specify STATEXIT and STATFLD.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the interval value in hundredths of seconds.

,LASTING=NO
,LASTING=YES

Use this input parameter to specify whether the member is to have permanent status recording. If you code LASTING=NO, XCF recognizes the member only while the member is active. If the sysplex is in XCF-local mode, you must use LASTING=NO.

If you code LASTING=YES, XCF recognizes the member even while the member is in the failed or quiesced state. If you are activating a member that already has permanent status recording in effect, use LASTING=YES.

MEMNAME is required with LASTING=YES.

,MEMASSOC=TASK
,MEMASSOC=JOBSTEP
,MEMASSOC=ADDRSPACE

Use this input parameter to specify a member's association with a task, job step task, or address space. XCF uses the MEMASSOC value to determine when to terminate a member (put the member in the failed or not-defined state). For more information on member termination, see *z/OS MVS Programming: Sysplex Services Guide*.

If you code MEMASSOC=TASK, the active member is associated with the task under which IXCJOIN was issued. XCF terminates the member when the associated task terminates.

If you code MEMASSOC=JOBSTEP, the active member is associated with the job step task under which IXCJOIN was issued. XCF terminates the member when the associated job terminates.

If you code MEMASSOC=ADDRSPACE, the active member is associated with the address space under which IXCJOIN was issued. XCF terminates the member when the associated address space terminates. Note that with this option, the system cannot provide SRB to task percolation.

Remember: An initiator address space does not end when a batch job running in it ends. So, if you are issuing IXCJOIN from a batch job and you want to terminate the member when the batch job ends, you must associate the member with the job, (MEMASSOC=JOBSTEP) not the initiator address space.

,MEMDATA=NO_MEMDATA
,MEMDATA=*memdata*

Use this input parameter to specify a 64-bit member data field. XCF provides this field to the group, status, message, and message notify user routines for this member. If you do not specify MEMDATA, or you specify MEMDATA=NO_MEMDATA, XCF sets the member data field to zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains member data.

,MEMNAME=NO_MEMNAME
,MEMNAME=*memname*

Use this input parameter to specify the name you choose for the member. If the member was previously in a created, failed, or quiesced state, you already have defined the member name. The name must be 16 characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, and national characters (@, # and $).

If you use LASTING=YES, MEMNAME is required (MEMNAME=NO_MEMNAME is not acceptable). If you are defining one member per system, consider using the SYSNAME from CVTSNAME as the member name.

If you do not specify MEMNAME, or if you specify MEMNAME=NO_MEMNAME, XCF generates the member name in the form Mxxxxxxxxxxxxxxx.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the XCF member name.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MSGEXIT=NO_MSGEXIT
,MSGEXIT=*msgexit*
Use this input parameter to specify the address of the message user routine. This routine executes in the primary address space of the issuer of the IXCJOIN macro, in 31-bit addressing mode and in SRB mode. The routine receives control when a message becomes available for this member from another member of the group.

See *z/OS MVS Programming: Sysplex Services Guide* for information about how to code a message user routine.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the message exit routine.

,MSGOUTASID=MEMBER
,MSGOUTASID=ANY
Use this input parameter to indicate which address spaces can use the IXCMSGO macro to send messages to the members in the specified XCF group.

MEMBER
Indicates that when IXCMSGO is issued, the primary address space must equal the requesting member's primary address space at the time the group was joined, or the primary address space must be the MASTER address space.

ANY
Indicates that IXCMSGO can be issued from any address space.

,NOTIFYEXIT=NO_NOTIFYEXIT
,NOTIFYEXIT=*notifyexit*
Use this input parameter to specify the name of a user routine to receive control for message notifications. The routine must reside in the user's address space and run in 31-bit addressing mode. See *z/OS MVS Programming: Sysplex Services Guide* for the types of events that the system can present to the message notify user routine.

Specifying or defaulting to NO_NOTIFYEXIT indicates that the member does not want to receive unsolicited notification of events that have occurred. Lack of a message notify user routine at join time does not prevent the member from specifying the name of a routine to other services (such as IXCMSGO) that support it.

See *z/OS MVS Programming: Sysplex Services Guide* for information about how to code a message notify user routine.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the address of the message notify user routine.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXCJOIN Version Support" on page 60 for a description of the options available with PLISTVER.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the join request is complete.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the join request completes.

,STATEXIT=*statexit*

Use this input parameter to specify the address of the status user routine. This routine executes in the primary address space of the issuer of the IXCJOIN macro, in 31-bit addressing mode and in SRB mode. The routine receives control if the status field of the member is unchanged during the time period determined by the INTERVAL parameter. If you specify STATEXIT, you must also specify STATFLD and INTERVAL.

See *z/OS MVS Programming: Sysplex Services Guide* for information about how to code a status user routine.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the status user-routine.

,STATFLD=NO_STATFLD
,STATFLD=*statfld*

Use this input parameter to specify a 64-bit status field in fixed or disabled reference (DREF) common storage, with any storage key. If the status field remains unchanged over the specified interval, XCF schedules the status user-routine identified in STATEXIT. If you specify STATFLD, you must also specify STATEXIT and INTERVAL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the status of the member.

,SYSCLEANUPMEM=NO
,SYSCLEANUPMEM=YES

Use this input parameter to specify whether the member will perform system-wide cleanup after a system leaves the sysplex. If YES is specified, the system will wait for this member to issue the IXCSYSCL macro before any elements may be restarted by automatic restart management.

NO

Indicates the member joining the group will not perform system-wide cleanup of resources when a system leaves the sysplex.

YES
Indicates the member joining the group will perform system-wide cleanup when a system leaves the sysplex. When this member is notified that a system has left the sysplex, it must issue the IXCSYSCL macro to indicate to the system that cleanup is complete.

,USLEN=*uslen*
Use this input parameter to specify the length in bytes of the user state data that you provide on the USTATE parameter. The length must be from 1 to 32 bytes. If you specify less than 32 bytes, XCF pads the remainder of the user state field, up to 32 bytes, on the right with zeros. IXCQUERY always returns the full 32 bytes, and group user-routines always receive the full 32 bytes. If you code USTATE, USLEN is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the user state data (USTATE).

,USTATE=NO_USTATE
,USTATE=*ustate*
Use this input parameter to specify the user state data that XCF is to place in the member's user state field. If you do not specify the USTATE parameter, XCF retains the existing value in the user state field unless the joining member was previously not-defined. In this case, XCF clears the user state field to zeros. Specify the length of the user state data on the USLEN parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of USLEN) that contains the user state information.

ABEND Codes

None.

Return and Reason Codes

When the IXCJOIN macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 6. Return and Reason Codes for the IXCJOIN Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** IXCJOIN completed successfully. XCF places the member in the active state, and places member information in ANSAREA. **Action:** None.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
04	04	**Equate Symbol:** IXCJOINRSNFIRSTACTIVEMEMBER **Meaning:** IXCJOIN completed successfully; the member is the first active member of the group. XCF places member information in ANSAREA. **Action:** None required.
04	08	**Equate Symbol:** IXCJOINRSNWASFAILED **Meaning:** IXCJOIN completed successfully; the member already exists in a failed state, and the caller correctly specified LASTING=YES. A new member token has been assigned to the new instance of the member. **Action:** None required. However, because the previous instance of the member was in a failed state, your application might need to perform some type of cleanup or takeover processing. MVS placed the member in the failed state when the member's corresponding task, address space, or system terminated. If the user state of the previous instance is needed and you must specify a user state on the IXCJOIN request, you can change your program to issue an IXCQUERY prior to the IXCJOIN to capture the user state of the previous instance. If you do not specify a user state on IXCJOIN, the new instance of the member will retain the user state of the old instance.
04	0C	**Equate Symbol:** IXCJOINRSNWASQUIESCED **Meaning:** IXCJOIN completed successfully; the member already exists in a quiesced state, and the caller correctly specified LASTING=YES. A new member token has been assigned to the new instance of the member. **Action:** None required. However, because the previous instance of the member was in a quiesced state, your application might need to perform some type of cleanup or takeover processing. The previous instance of the member voluntarily placed itself in a quiesced state. If the user state of the previous instance is needed and you must specify a user state on the IXCJOIN request, you can change your program to issue an IXCQUERY prior to the IXCJOIN to capture the user state of the previous instance. If you do not specify a user state on IXCJOIN, the new instance of the member will retain the user state of the old instance.
04	10	**Equate Symbol:** IXCJOINRSNWASCREATED **Meaning:** IXCJOIN completed successfully. The member already exists in a created state, and the caller correctly specified LASTING=YES. **Action:** None required. However, you might take some action based on your application. You can use the member's current user state to determine what action needs to be taken.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	04	**Equate Symbol:** IXCJOINRSNISCREATED **Meaning:** Program error. The member already exists in a created state, and the caller did not specify LASTING=YES. LASTING=YES must be specified since the member already has permanent status recording. The member's state and user state are not altered. **Action:** Ensure that your program specifies LASTING=YES, and retry the request. If permanent status recording is not required for this instance of the member, issue IXCDELET to delete the member, and retry the IXCJOIN request.
08	08	**Equate Symbol:** IXCJOINRSNISACTIVE **Meaning:** Program error. The member already exists in an active state. **Action:** None required. However, you might take some action depending on your application.
08	0C	**Equate Symbol:** IXCJOINRSNISQUIESCED **Meaning:** Program error. The member already exists in a quiesced state, and the caller did not specify LASTING=YES. LASTING=YES must be specified since the member already has permanent status recording. The member's state and user state are not altered. **Action:** Ensure that your program specifies LASTING=YES, and retry the request. If permanent status recording is not required for this instance of the member, issue IXCDELET to delete the member, and retry the IXCJOIN request. Before you delete the member, you can use the IXCQUERY service to capture the member's current user state.
08	10	**Equate Symbol:** IXCJOINRSNISFAILED **Meaning:** Program error. The member already exists in a failed state, and the caller did not specify LASTING=YES. **Action:** Ensure that your program specifies LASTING=YES, and retry the request. If permanent status recording is not required for this instance of the member, issue IXCDELET to delete the member, and retry the IXCJOIN request. Prior to deleting the member, you can use the IXCQUERY service to capture the member's current user state.
08	14	**Equate Symbol:** IXCJOINRSNGRPNAMEBAD **Meaning:** Program error. The group name is not valid. **Action:** Correct the group name, and retry the request.
08	18	**Equate Symbol:** IXCJOINRSNMEMNAMEBAD **Meaning:** Program error. The member name is not valid. **Action:** Correct the member name, and retry the request.
08	1C	**Equate Symbol:** IXCJOINRSNINTERVALBAD **Meaning:** Program error. The status-checking interval value is zero or is not a multiple of 100. **Action:** Correct the status-checking interval, and retry the request.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	20	**Equate Symbol:** IXCJOINRSNSTATFLDBADSTG **Meaning:** Program error. XCF could not access the STATFLD. **Action:** Ensure that the STATFLD parameter is correct. The STATFLD must be a doubleword in fixed or DREF common storage.
08	24	**Equate Symbol:** IXCJOINRSNLASTINGNEEDSMEMNAME **Meaning:** Program error. The caller specified LASTING=YES without specifying the member name. XCF will not generate member names for members that have permanent status recording. **Action:** If the member requires permanent status recording, provide a member name. If the member does not require permanent status recording, do not specify LASTING=YES.
08	28	**Equate Symbol:** IXCJOINRSNSTATUSMONINCOMPLETE **Meaning:** Program error. The STATFLD, INTERVAL, or STATEXIT parameter is missing. For XCF to monitor the member's status, all three of these values must be specified. **Action:** Ensure that the STATFLD, INTERVAL, and STATEXIT are correct, and retry the request.
08	xxyy003C	**Equate Symbol:** IXCJOINRSNANSAREAINCOMPLETE **Meaning:** Program error. The caller specified ANSAREA or ANSLEN incorrectly. Even though the member might have been placed in an active state, the ANSAREA was nonaddressable or the ANSLEN was not large enough. **Action:** Check the two high-order bytes of this reason code fullword *xxyy*003C for the return code *xx* (either 00 or 04) and reason code *yy* the caller would have received from IXCJOIN if the caller had coded those parameters correctly. Take action as described by the corresponding return and reason code. You might want to abnormally end your program or take some other action that will record the problem. You should correct your program to ensure that the ANSAREA is addressable and that the ANSLEN is large enough. See the description of these keywords for their requirements. If the return and reason code indicate that the member was placed in an active state, you can use the IXCQUERY service to get the information that would have been returned in the ANSAREA. **Note:** You should specify a member name or a unique user state value on IXCJOIN invocation. If you request to have XCF define the member name, you might not be able to use IXCQUERY to determine which member was created on your behalf.
08	40	**Equate Symbol:** IXCJOINRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	44	**Equate Symbol:** IXCJOINRSNMEMASSOCBAD **Meaning:** Program error. The member association value (task, address space or job) is not correct in the parameter list. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	100	**Equate Symbol:** IXCJOINRSNPLISTBADALET **Meaning:** Program error. Your program is not running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCJOIN macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET).
08	104	**Equate Symbol:** IXCJOINRSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCJOINRSNPLISTBADFUNCTION **Meaning:** Program error or environmental error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	10C	**Equate Symbol:** IXCJOINRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR ASC mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCJOIN macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	110	**Equate Symbol:** IXCJOINRSNUSTATEBADSTG **Meaning:** Program error. XCF could not access the USTATE value. **Action:** Take one or more of the following actions: • Ensure that the correct USTATE address was specified. • If your program is running in AR ASC mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCJOIN macro. – Ensure that the USTATE ALET corresponds to the parameter list address. • Ensure that the USTATE storage area was not inadvertently freed by your program.
08	114	**Equate Symbol:** IXCJOINRSNUSLENBADVALUE **Meaning:** Program error. The USLEN value is less than 1 or greater than 32. **Action:** Correct the USLEN, and retry the request.
08	118	**Equate Symbol:** IXCJOINRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCJOIN only while in task mode.
08	11C	**Equate Symbol:** IXCJOINRSNNOTENABLED **Meaning:** The caller is not enabled. **Action:** Correct your program so that it does not issue IXCJOIN while it is disabled.
08	120	**Equate Symbol:** IXCJOINRSNPRIMARYNOTHOME **Meaning:** Program error. The primary address space is not equal to the home address space. **Action:** Correct your program so that it does not use IXCJOIN while in cross memory mode. You might want to pass this restriction on to your caller when you are unsure of the environment in which your caller may call your program.
08	128	**Equate Symbol:** IXCJOINRSNTASKTERM **Meaning:** Program error. XCF does not allow the JOIN process during or after task termination. **Action:** Correct your program so that it does not issue IXCJOIN from a task that is terminating. Ensure that your application does not invoke IXCJOIN from a task termination resource manager. You might want to pass this restriction on to your caller when you are unsure of the environment in which your caller may call you.

Table 6. Return and Reason Codes for the IXCJOIN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0C	04	**Equate Symbol:** IXCJOINRSNMAXGROUPS **Meaning:** Environmental error. The maximum number of groups already exists. **Action:** Retry the request at least once. If the problem persists, consult your system programmer to determine if the number of groups defined in the couple data set should be increased, or if the XCF limit on the number of groups has been reached. Your application should cover cases in which the couple dataset has no available room. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements.
0C	08	**Equate Symbol:** IXCJOINRSNMAXMEMBERS **Meaning:** Environmental error. The maximum number of members in the group already exists. **Action:** Retry the request at least once. If the problem persists, consult your system programmer to determine if the maximum number of members defined in the couple data set should be increased, or if the XCF limit on the number of members has been reached. Your application should cover cases in which no room is available within the group. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements.
0C	10	**Equate Symbol:** IXCJOINRSNPARTITIONING **Meaning:** Environmental error. The system is being removed from the sysplex, and XCF permanently suspends all requests to join a group on this system. **Action:** The action is dependent on your application. You might want to prepare your application for system termination. (See the chapter on XCF in *z/OS MVS Programming: Sysplex Services Guide*)
0C	14	**Equate Symbol:** IXCJOINRSNXCFLOCALMODE **Meaning:** Environmental error. IXCJOIN with LASTING=YES is not allowed when the sysplex is in XCF-local mode. Permanent status recording requires a couple data set in which XCF can retain the member's status. XCF-local mode does not support a couple data set. **Action:** This depends on your program. If XCF-local mode is not supported by your application, you must make the system programmer aware of this requirement. The system programmer will have to ensure that the system is not IPLed in XCF-local mode. Before running or installing your application, you should make the system programmer aware of your XCF group and member resource requirements. (See *z/OS MVS Programming: Sysplex Services Guide*.)
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code and reason code, if applicable, and supply them to the appropriate IBM support personnel.

Example

Operation: Join a member MEMB1 to a group MYGROUP with LASTING=NO. Place X'11' in the user state field, and assign an interval of 1000 (or 10 seconds). Register 2 points to the area where the member information will be returned. The user-routine addresses are as follows:

- Group user-routine address is in register 4.
- Status user-routine address is in register 5.
- Message user-routine address is in register 6.

The status field must be in common storage. Register 7 has the address of this area, obtained through the STORAGE OBTAIN macro. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         LA    R2,MYAREA                OBTAIN ADDRESS OF OUTPUT AREA  X
                                        FOR IXCJOIN
         L     R4,GXTADDR               OBTAIN ADDRESS OF GROUP        X
                                        USER-ROUTINE FOR IXCJOIN
         L     R5,SXTADDR               OBTAIN ADDRESS OF STATUS       X
                                        USER-ROUTINE FOR IXCJOIN
         L     R6,MXTADDR               OBTAIN ADDRESS OF MESSAGE      X
                                        USER-ROUTINE FOR IXCJOIN
         STORAGE OBTAIN,LENGTH=8,SP=228  OBTAIN STORAGE FOR STATUS     X
                                         FIELD
         ST    R1,FIELD1                SAVE ADDRESS OF STATUS FIELD
         LR    R7,R1                    PLACE ADDRESS IN REGISTER FOR  X
                                        IXCJOIN INVOCATION


         IXCJOIN  GRPNAME=MYGROUP,ANSAREA=(R2),ANSLEN=AREALEN,         X
               LASTING=NO,MEMNAME=MEMB1,STATFLD=(R7),                  X
               MEMASSOC=JOBSTEP,
               GRPEXIT=(R4),STATEXIT=(R5),MSGEXIT=(R6),                X
               MEMDATA=DATA1,INTERVAL=INTER1,                          X
               USTATE=STATE1,USLEN=LEN,                                X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

         EXTRN GEXIT
         EXTRN SEXIT
         EXTRN MEXIT

MYGROUP  DC    CL8'MYGROUP '            GROUP NAME
MYAREA   DS    CL124                    OUTPUT AREA TO CONTAIN DATA    X
                                        RETURNED BY IXCJOIN
DATA1    DS    CL8                      MEMBER DATA FOR THIS MEMBER
FIELD1   DS    1F                       ADDRESS OF STATUS FIELD
RETURN   DS    1F                       RETURN CODE
REASON   DS    1F                       REASON CODE
STATE1   DC    X'11'                    USER STATE VALUE
LEN      DC    F'1'                     LENGTH OF USER STATE DATA
INTER1   DC    F'1000'                  INTERVAL VALUE
GXTADDR  DC    A(GEXIT)                 ADDRESS OF GROUP USER-ROUTINE
SXTADDR  DC    A(SEXIT)                 ADDRESS OF STATUS USER-ROUTINE
MXTADDR  DC    A(MEXIT)                 ADDRESS OF MESSAGE USER-ROUTINE
AREALEN  DC    F'124'                   LENGTH OF OUTPUT AREA
MEMB1    DC    CL16'MEMB1           '   MEMBER NAME
```

IXCLEAVE — Place an XCF Member in the Not-Defined State

Description

The IXCLEAVE macro allows a member of a cross-system coupling facility (XCF) group to change its state from active to not-defined. XCF does not recognize the existence of a member that is in a not-defined state.

Additionally, IXCLEAVE notifies active members in the group that have defined a group user-routine that the target member is now in the not-defined state. XCF delivers outstanding messages sent by the member and discards undelivered messages that were sent to the member.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any HASN, any SASN. The primary address space must be the same as the primary address space of the caller of the IXCJOIN that placed the member in the active state, or the caller must be executing in the master scheduler address space.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in access register (AR) mode, issue the SYSSTATE ASCENV=AR macro before IXCLEAVE. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

The caller can have no enabled, unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCLEAVE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

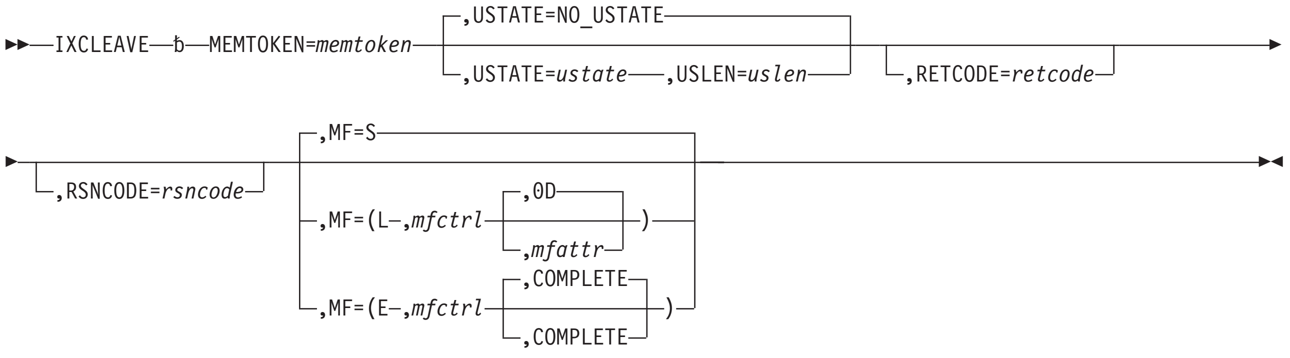
Performance Implications

None.

Syntax Diagram

The syntax of the IXCLEAVE macro is as follows:

IXCLEAVE diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

MEMTOKEN=*memtoken*

Use this input parameter to specify the 64-bit token of the member that IXCLEAVE is to place in the not-defined state. XCF provided this token when it activated the member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token that was returned in the ANSAREA when IXCJOIN completed.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,USLEN=*uslen*
Use this input parameter to specify the length in bytes of the user state data that you provide on the USTATE parameter. The length must be from 1 to 32 bytes. XCF overlays any previous value in the user state field up to the length you specify on USLEN. XCF does not pad the remainder of the user state field (up to 32 bytes) with zeros. IXCQUERY always returns the full 32 bytes, and group user-routines always receive the full 32 bytes. If you code USTATE, USLEN is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the user state data (USTATE).

,USTATE=NO_USTATE
,USTATE=*ustate*
Use this input parameter to specify the area containing data that you want XCF to place in the user state field associated with the member. Use the USLEN parameter to specify the length of the user state data.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) that contains the user state information.

ABEND Codes

None.

Return and Reason Codes

When the IXCLEAVE macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 7. Return and Reason Codes for the IXCLEAVE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** IXCLEAVE completed successfully. XCF places the member in the not-defined state and informs group members of the change. **Action:** None.
04	04	**Equate Symbol:** IXCLEAVERSNEXITSNOTPURGED **Meaning:** Environmental error. IXCLEAVE completed successfully. XCF did not purge the group, status, or message user-routine SRBs successfully. For a group, status, or message exit, it is possible that the member specified on IXCJOIN is still running. Once an exit returns to XCF, it will not be scheduled again. **Action:** Your application should be aware that one of the exits might still be running. If this return code occurs more than once for IXCLEAVE, take the following actions: • Determine if any asynchronous abends are being issued against the current task. If so, you may need to reduce their frequency. • XCF should have taken an SDUMP to record the abend or abends. If the SDUMP indicates that the error was caused by SRB-to-task percolation or application code issuing a CALLRTM against your task, this is probably not an XCF error. If you feel this is an XCF error, record the return and reason code, and supply it to the appropriate IBM support personnel.
08	04	**Equate Symbol:** IXCLEAVERSNNOTACTIVE **Meaning:** Program error. The member token does not identify an active member. **Action:** Ensure that the correct MEMTOKEN was specified. Further action depends on your application. If the member must be in a not-defined state, you can use the IXCQUERY service to determine what state the member is currently in. If the member is currently in a failed, quiesced, or created state, you can use the IXCDELET service to place the member in a not-defined state.

Table 7. Return and Reason Codes for the IXCLEAVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	08	**Equate Symbol:** IXCLEAVERSNINAPPROPRIATEPRIMARY **Meaning:** Program error. The primary address space is neither the master scheduler address space nor the primary address space of the caller that issued IXCJOIN to place the member in the active state. **Action:** Ensure that: • The correct MEMTOKEN was specified. • Your program issues IXCLEAVE only from the master scheduler address space or the address space from which the member (MEMTOKEN) joined the group.
08	10	**Equate Symbol:** IXCLEAVERSINAPPROPRIATESYSTEM **Meaning:** Program error. The system is not the system on which the IXCJOIN for the member was issued. **Action:** Ensure that: • The correct MEMTOKEN was specified. • Your program issues IXCLEAVE only for members that joined on the same system as your program.
08	40	**Equate Symbol:** IXCLEAVERSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	100	**Equate Symbol:** IXCLEAVERSNPLISTBADALET **Meaning:** Program error. Your program is running in AR mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR prior to issuing the IXCLEAVE macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET). • Your program is not intended to run in primary ASC mode.
08	104	**Equate Symbol:** IXCLEAVERSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCLEAVERSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.

Table 7. Return and Reason Codes for the IXCLEAVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	10C	**Equate Symbol:** IXCLEAVERSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCLEAVE macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
08	110	**Equate Symbol:** IXCLEAVERSNUSTATEBADSTG **Meaning:** Program error. XCF could not access the USTATE value. **Action:** Take one or more of the following actions: • Ensure that the correct USTATE address was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCLEAVE macro. – Ensure that the USTATE ALET is correct. • Ensure that the USTATE storage area was not inadvertently freed by your program.
08	114	**Equate Symbol:** IXCLEAVERSNUSLENBADVALUE **Meaning:** Program error. The USLEN value is less than 1 or greater than 32. **Action:** Correct the USLEN, and retry the request.
08	118	**Equate Symbol:** IXCLEAVERSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCLEAVE only while in task mode.
08	11C	**Equate Symbol:** IXCLEAVERSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCLEAVE only while enabled.
08	120	**Equate Symbol:** IXCLEAVERSNPRIMARYNOTHOME **Meaning:** Program error. The primary address space is not equal to the home address space. **Action:** Correct your program so that it does not use IXCLEAVE while in cross memory mode. You might want to pass this restriction on to your caller when you are unsure of the environment your caller may have been in.

Table 7. Return and Reason Codes for the IXCLEAVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	18	**Equate Symbol:** IXCLEAVERSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended (i.e. another unit of work attempted to abnormally terminate this task). The state of the IXCLEAVE request is unpredictable. **Action:** Determine why another unit of work has decided to abnormally terminate this task.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

A member places itself in the not-defined state. XCF is to place the value X'11' in the user state field and store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCLEAVE MEMTOKEN=TOKEN1,USTATE=STATE,USLEN=LEN,             X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

TOKEN1   DS    CL8                      TOKEN OF MEMBER TO BE PLACED   X
                                        IN NOT-DEFINED STATE
RETURN   DS    1F                       RETURN CODE
REASON   DS    1F                       REASON CODE
STATE    DC    X'11'                    USER STATE VALUE
LEN      DC    F'1'                     LENGTH OF USER STATE DATA
```

You can obtain the member token from the QUAMTOKN field in the area returned by IXCJOIN or IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXCMG — Obtain Tuning and Capacity Planning Data

Description

The IXCMG macro provides a cross-system coupling facility (XCF) installation with information that can help system programmers plan tuning activities and capacity requirements for the sysplex.

The data that IXCMG generates consists of one header, followed by zero or more data records. The IXCYAMDA mapping macro maps the data that IXCMG returns, including the header that the AMDAREA structure maps.

The TYPE parameter specifies which of four types of data records you can request:

- TYPE=PATH returns one record for each signalling path that XCF is using.
- TYPE=MSGPEND returns one record for each pending or delayed message.
- TYPE=SYSTEM returns records summarizing message traffic for the system.
- TYPE=SRCDST returns counts of messages sent and received by members in the system.

TYPE=ALL, the default, returns all possible types of data records.

Two required parameters, DATAAREA and DATALEN, specify the area where IXCMG is to return the data.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Holds either no locks or only the CPU lock
Control parameters:	Must be in the primary address space or addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCMG. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Include the mapping macro IXCYAMDA in your program to map the data returned in DATAAREA.

Restrictions

This macro must be issued from a nonswappable primary address space.

Input Register Information

Before issuing the IXCMG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register **Contents**

0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the macro; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

The use of the IXCMG macro might degrade the performance of the XCF signalling service.

Understanding IXCMG Version Support

The IXCMG macro supports versions in the range of 0 - 1.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXCMG macro.
- The following keywords are supported by all versions starting with version 1 and higher of the IXCMG macro.

AMDALEVEL
MEMTOKEN
TYPE=MEMBER

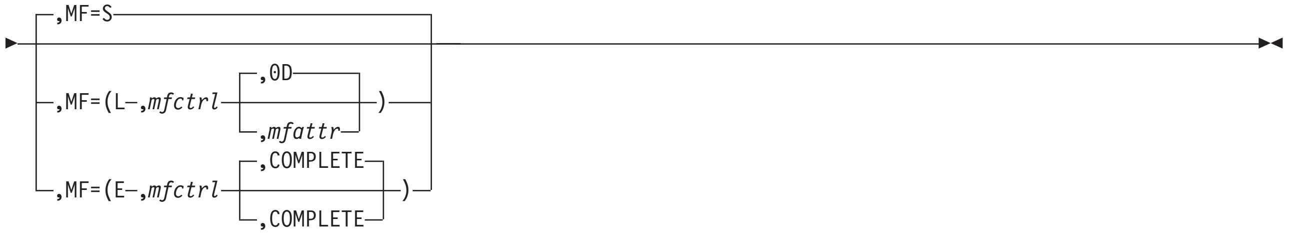
Specify the version of the parameter list that you want generated with the PLISTVER keyword. See "Specifying a Macro Version Number" on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCMG macro is as follows:

IXCMG diagram

```
                                                   ┌─,TYPE=ALL──────┐
►►──IXCMG──␢──DATAAREA=dataarea──,DATALEN=datalen──┼────────────────┼──┬────────────────────┬──────────►
                                                   ├─,TYPE=PATH─────┤  ├─,MEMTOKEN=0────────┤
                                                   ├─,TYPE=MSGPEND──┤  └─,MEMTOKEN=memtoken─┘
                                                   ├─,TYPE=SYSTEM───┤
                                                   ├─,TYPE=SRCDST───┤
                                                   └─,TYPE=MEMBER───┘

►──┬──────────────────────┬──┬───────────────────┬──┬───────────────────┬──────────────────────────►◄
   ├─,AMDALEVEL=0─────────┤  └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘
   └─,AMDALEVEL=amdalevel─┘
```



Note: To specify more than one TYPE, enclose the choices in parenthesis and separate them by commas. For example, TYPE=(PATH,SYSTEM).

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,AMDALEVEL=0
,AMDALEVEL=*amdalevel*

Use this input parameter to specify the level of the IXCYAMDA record mappings that the system returns in the answer area. Valid values are 0 and 1.

- A value of 0 indicates that base level IXCYAMDA records will be returned.
- A value of 1 indicates that level-1 IXCYAMDA records will be returned.

DATAAREA=*dataarea*

Use this output area to identify the area in which IXCMG is to return the data you request.

The data area must be in fixed or disabled reference (DREF) storage. It must be in the primary address space or in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) that identifies the data area.

,DATALEN=*datalen*

Use this input parameter to specify the length of the area you provided on the DATAAREA parameter. The length should be long enough to accommodate the IXCYAMDA mapping of the data area.

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) that contains the length of the data area.

,MEMTOKEN=0
,MEMTOKEN=*memtoken*

Use this input parameter to specify the MEMTOKEN of the member whose MEMBER data is to be gather. MEMTOKEN is only used when gathering MEMBER data. It is ignored for all other data options. If the indicated member does not reside on the local system, no MEMBER data will be returned.

To Code: Specify the name of a 64-bit field (or register 2-12 containing the address of the field) that contains the MEMTOKEN of the member.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) to contain the reason code.

,TYPE=ALL
,TYPE=PATH
,TYPE=MSGPEND
,TYPE=SYSTEM
,TYPE=SRCDST
,TYPE=MEMBER
Use this input parameter to specify which types of data records XCF is to return. TYPE=ALL returns all types of data records. If you do not specify the TYPE parameter, you receive all types of data records. You may specify more than one type of data record with this parameter. For example, you could code: TYPE=(PATH,MSGPEND).

The data that IXCMG generates consists of a header, followed by zero or more data records. The types of data records are as follows:

- TYPE=PATH returns one record for each signalling path that XCF is using. The AMDPATH structure in IXCYAMDA maps this record. It includes such information as, for each outbound signalling path, the number of times XCF tried to send a signal across the path.
- TYPE=MSGPEND returns one record for each pending message. The AMDMPEND structure maps this record. It includes such information as, for each outbound signal queued for data transfer, the member token, ASID, and length of the message.
- TYPE=SYSTEM returns records that describe the message traffic associated with the system on which you issue IXCMG. The AMDSYS structure maps this record. It includes such information as the total number of times the system refused message requests in each transport class because buffer space was not available.
- TYPE=SRCDST returns one record for each active member in the sysplex. The AMDSD structure maps this record. It includes such information as the number of messages the member has sent.
- TYPE=MEMBER returns one record for each member that resides on the local system. The AMDMEM structure maps this information. The MEMTOKEN keyword can be specified to limit the data to a specific member.

ABEND Codes

None.

Return and Reason Codes

When the IXCMG macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 8. Return and Reason Codes for the IXCMG Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** IXCMG completed successfully, and XCF provided all the data that the caller requested. **Action:** None.

Table 8. Return and Reason Codes for the IXCMG Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	4	**Equate Symbol:** IXCMGRSNSTILLMOREDATA **Meaning:** Program error. IXCMG completed successfully, and XCF provided some data; however, DATAAREA is too small to contain all the requested data. **Action:** Take one or more of the following actions: • If the required information is contained in the DATAAREA, no further action needs to be taken. • If the required information is not contained in the DATAAREA, obtain a larger DATAAREA and reissue IXCMG. The AMDATLEN field, as defined by the XCF accounting and measurement data area mapping macro (IXCYAMDA), contains the DATAAREA size that is required to contain all of the information that would have been returned. However, because IXCMG returns only a snapshot of the current status, it is possible that the AMDATLEN might be too small on the next invocation. • Ensure that you specified the correct length for the data area. • Ensure that the parameter list was not inadvertently overlaid. • Ensure that you specified the correct data area address and ALET. • If your program is not running in primary ASC mode, ensure that you specified SYSSTATE ASCENV=AR.
8	14	**Equate Symbol:** IXCMGRSNDATAAREATOOSMALL **Meaning:** Program error. DATAAREA field is too small to contain even the header (AMDAREA). **Action:** Take one or more of the following actions: • Provide a data area that is large enough to contain at least the header information as mapped by the field AMDAREA of the XCF accounting and measurement data area mapping macro (IXCYAMDA). • Ensure that you specified the correct length for the data area. • Ensure that the parameter list was not inadvertently overlaid. • Ensure that you specified the correct data area address and ALET. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR.
8	18	**Equate Symbol:** IXCMGRSNDATAAREABADSTG **Meaning:** Program error. XCF could not access DATAAREA. **Action:** Take one or more of the following actions: • Ensure that the correct DATAAREA address, ALET, and length were specified. • Ensure that the parameter list storage area was not inadvertently freed by your program. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCMG macro.

Table 8. Return and Reason Codes for the IXCMG Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	1C	**Equate Symbol:** IXCMGRSNBADALET **Meaning:** Program error. The ALET that qualifies the address specified on DATAAREA is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • If your program is running in AR mode, you specified SYSSTATE ASCENV=AR before issuing the IXCMG macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET).
8	40	**Equate Symbol:** IXCMGRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. Your program might have inadvertently written over an area in the control parameter list. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
8	100	**Equate Symbol:** IXCMGRSNPLISTBADALET **Meaning:** Program error. Your program is not running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCMG macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET).
8	104	**Equate Symbol:** IXCMGRSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
8	108	**Equate Symbol:** IXCMGRSNPLISTBADFUNCTION **Meaning:** Program error or environmental error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 8. Return and Reason Codes for the IXCMG Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	10C	**Equate Symbol:** IXCMGRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCMG macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Operation: Obtain tuning and capacity planning information on the use of XCF signalling paths. Register 2 points to the area where XCF is to return member information. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         LA    R2,MYAREA     LOAD ADDRESS OF DATA AREA

         IXCMG DATAAREA=(R2),DATALEN=AREALEN,TYPE=PATH,                X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

MYAREA   DS    CL4096        AREA TO PLACE MEASUREMENT DATA
RETURN   DS    1F            RETURN CODE
REASON   DS    1F            REASON CODE
AREALEN  DC    F'4096'       LENGTH OF THE DATA AREA WHERE             X
                             MEASUREMENT INFORMATION IS PLACED
```

IXCMOD — Modify Status-Checking Interval

Description

The IXCMOD macro allows an active cross-system coupling facility (XCF) member to change its status-checking interval. The member must have status monitoring established.

A typical use of IXCMOD is for the caller to change its interval in response to a change in the system's environment, such as a change in the failure detection interval. See *z/OS MVS Programming: Sysplex Services Guide* for further information.

When a member changes its status-checking interval through IXCMOD, XCF notifies other active members of the group through their group user-routines.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space of the caller of the IXCJOIN that placed the member in the active state.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for external and I/O interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space addressable through a public entry in the dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCMOD. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

- The member must have status monitoring established.
- The caller can have no enabled unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCMOD macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

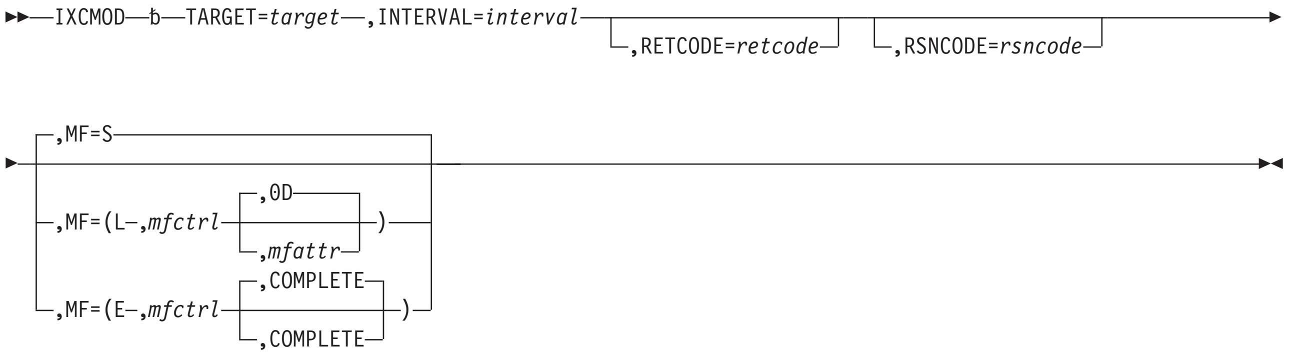
Performance Implications

When a member fails to update its status field within the member's status-checking interval (the INTERVAL parameter on the IXCJOIN or the IXCMOD macro), XCF schedules the member's status user-routine. By increasing the interval value, you decrease the number of times the status user-routine receives control, thereby reducing system overhead.

Syntax Diagram

The syntax of the IXCMOD macro is as follows:

IXCMOD diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,INTERVAL=*interval*

Use this input parameter to specify the status-checking interval that is to replace the existing status-checking interval. IXCJOIN set this interval on the INTERVAL parameter, or IXCMOD reset the interval in a previous invocation. Specify the interval in full seconds; that is, the value must be greater than zero and must be a multiple of 100.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the new interval.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when this request completes.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request completes.

TARGET=*target*
Use this input parameter to specify the token of the target member. IXCJOIN provided this token when it activated the member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token.

ABEND Codes

None.

Return and Reason Codes

When the IXCMOD macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

IXCMOD Macro

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 9. Return and Reason Codes for the IXCMOD Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
00	None.	**Meaning:** IXCMOD completed successfully; XCF changed the value of the status-checking interval. **Action:** None.
08	04	**Equate Symbol:** IXCMODRSNNOTACTIVE **Meaning:** Program error. The target member is not active. **Action:** Take one or more of the following actions: • Ensure that the correct MEMTOKEN was specified. • You might want your program to take some action based on the fact the member is no longer active.
08	08	**Equate Symbol:** IXCMODRSNNOSTATUSMON **Meaning:** Program error. The IXCJOIN that activated the target member did not request status monitoring, or XCF has stopped monitoring the member. **Action:** Take one or more of the following actions: • Ensure that the correct MEMTOKEN was specified. Verify the address and ALET (if appropriate). • Correct your application so that it joins the member with status monitoring.
08	0C	**Equate Symbol:** IXCMODRSNINTERVALBAD **Meaning:** Program error. The status-checking interval value is either zero or not a multiple of 100. **Action:** Take one or more of the following actions: • Ensure that the correct status-checking interval was specified. Verify the address and ALET (if appropriate). • Correct your program and retry the request.
08	10	**Equate Symbol:** IXCMODRSNINAPPROPRIATECALLER **Meaning:** Program error. The primary address space is not the same as the primary address space of the issuer of the IXCJOIN that activated the target member. **Action:** Take one or more of the following actions: • Ensure that the correct MEMTOKEN was specified. Verify the address and ALET (if appropriate). • Correct your program so that it issues IXCMOD only from the address space in which the IXCJOIN was issued for the member.

Table 9. Return and Reason Codes for the IXCMOD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	40	**Equate Symbol:** IXCMODRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	100	**Equate Symbol:** IXCMODRSNPLISTBADALET **Meaning:** Program error. The ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • If your program is running in AR mode, you specified SYSSTATE ASCENV=AR before issuing the IXCMOD macro. • The ALET for the parameter list is a valid entry on the DU-AL or is zero (primary address space ALET).
08	104	**Equate Symbol:** IXCMODRSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCMODRSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.
08	10C	**Equate Symbol:** IXCMODRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCCREAT macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
08	118	**Equate Symbol:** IXCMODRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCMOD only while in task mode.

Table 9. Return and Reason Codes for the IXCMOD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	11C	**Equate Symbol:** IXCMODRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so it only calls IXCMOD while enabled.
0C	18	**Equate Symbol:** IXCMODRSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended (ie. another unit of work attempted to abnormally terminate this task). The state of the IXCMOD request is unpredictable. **Action:** Determine why the task was being abended.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Operation: A member changes its own status-checking interval to five seconds. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCMOD  TARGET=TOKEN1,INTERVAL=INTER,                         X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

RETURN   DS    1F                        RETURN CODE
REASON   DS    1F                        REASON CODE
TOKEN1   DS    CL8                       TOKEN OF THE MEMBER TO BE     X
                                         MODIFIED
INTER    DC    F'500'                    NEW INTERVAL VALUE FOR THIS   X
                                         MEMBER
```

You can obtain the member token from the QUAMTOKN field in the area returned by IXCJOIN or IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXCMSGC — XCF Message Control

Description

The IXCMSGC macro allows a cross-system coupling facility (XCF) application to interact with the XCF signalling services to provide additional functions. The IXCMSGC macro provides services that:
- Save a message for later processing
- Retrieve information about messages that have been saved or are pending completion
- Redeliver a message to the member that previously saved the message
- Force a message-out request to be considered complete
- Discard a message that a member had previously saved
- Cancel (discard) a pending message.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must equal the primary address space of the caller of the IXCJOIN macro when it was issued to join the XCF group.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCMSGC. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Include the mapping macro IXCYMQAA in your program to map the data returned in ANSAREA when using the IXCMSGC REQUEST=QUERYMSG service. Use the length and offsets provided in the IXCYMQAA records to ensure compatibility with additional data provided in the future.

Note that some request options are valid only when running in task mode, or when running as a message user routine or as a message notify user routine.

Restrictions

The virtual storage areas specified by ANSAREA and RETMSGTOKEN must be addressable in the caller's primary address space, in an address space or data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL), or in a common area data space.

A IXCMSGC REQUEST=CALLEXIT invocation cannot be made with FRRs established while in task mode.

Input Register Information

Before issuing the IXCMSGC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXCMSGC macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if GPR 15 contains a non-zero return code that has applicable reason codes.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers and restore them after the system returns control.

Understanding IXCMSGC Version Support

The IXCMSGC macro supports version 0.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCMSGC macro is as follows:

main diagram

```
►►──IXCMSGC──b──MEMTOKEN=memtoken──┬─,REQUEST=SAVEMSG──┤ parameters-1 ├───┬──┬──────────────────┬──►
                                   ├─,REQUEST=DISCARDMSG──,TOKEN=token─────┤  └─,RETCODE=retcode─┘
                                   ├─,REQUEST=QUERYMSG──┤ parameters-2 ├───┤
                                   ├─,REQUEST=COMPLETION──┤ parameters-3 ├─┤
                                   └─,REQUEST=CALLEXIT──┤ parameters-4 ├───┘

                      ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┬──────────────────┬──┼───────────────────────────┼──┼────────────────────────────────────────┼──►◄
   └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D──────┐          │
                         └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl─┼──────────┼─)──────┤
                                                         │                  └─,mfattr──┘          │
                                                         │                  ┌─,COMPLETE─┐         │
                                                         └─,MF=(E─,mfctrl─┼───────────┼─)─────┘
                                                                            └─,COMPLETE─┘
```

parameters-1

```
                    ┌─,USERDATA=NO_CHANGE─┐
►►──,TOKEN=token──┼─────────────────────┼──┬────────────────────────────┬──►◄
                    └─,USERDATA=userdata──┘  └─,RETMSGTOKEN=retmsgtoken─┘
```

parameters-2

```
►►──┬─,DATATYPE=MSGOUT──┬─,STATUS=SAVED──────┬─┬─,ANSAREA=ansarea─,ANSLEN=anslen─────────────►◄
    │                   ├─,STATUS=COMPLETE───┤ │
    │                   └─,STATUS=INCOMPLETE─┘ │
    │                  ┌─,SOURCE=ANY────┐      │
    ├─,DATATYPE=MSGIN──┼────────────────┼──────┤
    │                  └─,SOURCE=source─┘      │
    └─,DATATYPE=DETAIL──,TOKEN=token───────────┘
```

parameters-3

```
                  ┌─,USERDATA=NO_CHANGE─┐
►►─,TYPE=FORCE────┼─────────────────────┼─,TOKEN=token───────────────────────────►◄
                  └─,USERDATA=userdata──┘
```

parameters-4

```
                                       ┌─,EXITPARMS=0─────────┐
►►──┬─,NOTIFYEXIT=notifyexit─┬─────────┼──────────────────────┼─,TOKEN=token─────────►◄
    └─,MSGEXIT=msgexit───────┘         └─,EXITPARMS=exitparms─┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ANSAREA=*ansarea*

Use this output parameter to identify the storage area to contain the data returned by the REQUEST=QUERYMSG service. The data returned consists of a header record followed by zero or more records appropriate to the type of query. The IXCYMQAA macro defines the mappings for the header record and data records.

The storage area specified by ANSAREA must either be in the caller's primary address space or in an address or data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL) or in a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the output area where IXCMSGC is to return the requested information.

,ANSLEN=*anslen*

Use this input parameter to specify the length in bytes of the area you provided on the ANSAREA parameter.

The length of the answer area must be large enough to contain a complete header record (mapped by MQAHEADER). If the answer area is not large enough to contain all the available data records, data collection stops. The header record indicates how much storage would have been needed to collect all the data for the request (field MQAHDRTLEN). Note that the amount of storage needed to collect all the data can change by the time a new query is attempted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field, or a decimal value, that contains the length in bytes of the answer area.

,DATATYPE=MSGOUT
,DATATYPE=MSGIN
,DATATYPE=DETAIL

Use this input parameter to indicate what type of information is to be collected by the REQUEST=QUERYMSG request.

DATATYPE=MSGOUT

Collect summary information about messages sent by the member through the message-out service (IXCMSGO). The data returned for each message includes:
- A token that identifies the message
- User data associated with the message
- Status of the message.

This data is mapped by the MQAMSGOUTSUMMARY record defined in IXCYMQAA.

Note that a message reported by this query might not exist by the time the member making the query request attempts to use the message token returned in the answer area. For example, a message that was incomplete at the time of the query could complete and be processed by a message notify user routine before the member could use the message token returned from the query.

DATATYPE=MSGIN

Collect summary information about incoming messages that the member saved. Information is collected for messages that were saved by the message user routine or for responses saved by the message notify user routine. The data returned for each message includes:
- A token that identifies the message
- User data associated with the message
- The member token of the member that sent the message.

This data is mapped by the MQAMSGINSUMMARY record defined in IXCYMQAA.

DATATYPE=DETAIL

Collect detail information about the message identified by the TOKEN parameter. The data returned for the message depends on the type of message.
- For a message-out request, the data returned includes:
  - A token that identifies the message
  - User data associated with the message
  - Number of targets for the message
  - Message control data from the message-out request.
  - Flags to describe the characteristics of the message.
  - A table of response data with an entry for each possible target. The entry describes the result of the send and the associated response collection (if any). The entry also provides status information about the send, including whether it might require the asynchronous access of user storage.

  This data is mapped by the MQAMSGOUTDETAIL record defined in IXCYMQAA.
- For a message saved by the message user routine or for a response saved by the message notify user routine, the data returned includes:
  - A token that identifies the message
  - User data associated with the message
  - The member token of the member that sent the message
  - Message length
  - Message control data from the sender.

  This data is mapped by the MQAMSGINDETAIL record defined in IXCYMQAA.

,EXITPARMS=ZERO
,EXITPARMS=*exitparms*

Use this input parameter to specify the storage area that contains user parameters to be passed to the user routine identified either by the NOTIFYEXIT or MSGEXIT parameter of IXCMSGC.
- For a message user routine, MEPLEXEXITPARMS within the message exit parameter list (mapped by IXCYMEPL) contains a copy of these parameters.

- For a message notify user routine, MNPLEXITPARMS within the message notification parameter list (mapped by IXCYMNPL) contains a copy of these parameters.

The user can define the content and meaning of the 64-bit area. For example, you could use the area to pass the address and ALET of a storage area containing information that determines how the exit routine should perform its processing.

If the user does not specify EXITPARMS, the default is to set the parameters to X'0'.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 64-bit storage area containing the user parameters.

MEMTOKEN=*memtoken*

Use this input parameter to specify the member token that was returned by IXCJOIN and that identifies the member making the IXCMSGC request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-bit field that contains the member token.

,MSGEXIT=*msgexit*

Use this input parameter to identify the message user routine in the joiner's address space that is to receive control as specified by a REQUEST=CALLEXIT invocation. The user routine need not be the same as the message user routine defined when the member invoked IXCJOIN to join the XCF group.

Note that MSGEXIT is mutually exclusive with NOTIFYEXIT.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 31-bit message user routine that is to receive control.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,NOTIFYEXIT=*notifyexit*
Use this input parameter to identify the message notify user routine in the joiner's address space that is to receive control as specified by a REQUEST=CALLEXIT invocation. The user routine need not be the same as the message notify user routine defined when the member invoked IXCJOIN to join the XCF group nor need it be the one specified on the IXCMSGO invocation to send the message.

Note that NOTIFYEXIT is mutually exclusive with MSGEXIT.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 31-bit message notify user routine that is to receive control.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXCMSGC Version Support" on page 98 for a description of the options available with PLISTVER.

,REQUEST=SAVEMSG
,REQUEST=DISCARDMSG
,REQUEST=QUERYMSG
,REQUEST=COMPLETION
,REQUEST=CALLEXIT
Use this input parameter to indicate the type of request to be processed.

REQUEST=SAVEMSG
Save the message for later processing. This request is valid only when invoked from a message user routine or a message notify user routine. Use the TOKEN parameter to identify the message that is to be saved. If the user routine neither saves the message with the SAVEMSG service nor receives the message with the IXCMSGI service, the system automatically discards the message when the user routine gives up control.

A message notify user routine can save the entire message-out request including any available responses and can save individual responses as independent messages. A message user routine can save its message.

Once the user routine saves the message, the system invalidates the message token passed via the TOKEN keyword. Thus, the message is no longer available to the user routine because both the IXCMSGI service and the IXCMSGC service will reject requests that attempt to use the invalidated token. You can process a saved message by invoking the IXCMSGC REQUEST=CALLEXIT service.

The system automatically discards saved messages when the member becomes not active. An active member can discard its saved message by invoking the IXCMSGC REQUEST=DISCARDMSG service.

When a message is saved from within a message user routine, only the undelivered portion of the message is saved along with the data required to create a new message exit parameter list (MEPL). Once a member has received message data into the member's buffer area, the message data is considered delivered. The delivered part of the message cannot be redelivered (with

REQUEST=CALLEXIT) nor can it be saved. When a partially delivered message is saved, only the undelivered portion of the message is saved for later retrieval by the REQUEST=CALLEXIT service.

When a message with or without its associated responses is saved from within a message notify user routine, the message and its associated responses can be saved as a single entity or each individual response can be saved independently of the message or both.

- When saved as a single message/response entity, the undelivered portion of the responses is saved along with the data required to create a new message notification parameter list (MNPL). The token, passed by the TOKEN parameter, identifying the message is invalidated, as well as all the message tokens identifying the individual responses, if any, associated with the message (MNPLTRMSGITOKEN). The saved message/response entity can be processed at a later time by a message notify user routine specified on the IXCMSGC REQUEST=CALLEXIT service.
- When an individual response is saved as an independent message, the undelivered portion of the responses is saved along with the data required to create a new message exit parameter list (MEPL). The message token passed with the TOKEN parameter is invalidated. Note that a saved response is considered an independent message and must be processed as such with a message user routine specified on the IXCMSGC REQUEST=CALLEXIT service.

REQUEST=DISCARDMSG

Discard the message identified by the TOKEN parameter, thus making the discarded message no longer available for processing. For example, a discarded message cannot be presented to a user routine, nor can it be visible to the IXCMSGC REQUEST=QUERYMSG service. (Note however, that the IXCMSGC QUERYMSG service might return a message for which a discard is pending.) Any responses still associated with the discarded message are also discarded, except for saved responses. (Saving a response causes it to become disassociated from the message.)

If a message is neither explicitly saved by a user routine nor received by invoking the message-in service (IXCMSGI), the system automatically discards the message when the user routine gives up control. Thus, most members do not need to use the DISCARDMSG service from within a user routine when they elect not to process the message. The automatic discard is preferred over an explicit discard within a user routine because the automatic discard incurs less system overhead.

If a message is discarded while a user routine is processing that same message, subsequent attempts by the user routine to process the message with XCF services are rejected with a return and reason code indicating that the message has been discarded. Depending on the timing, the current operation being performed by the user routine might be allowed to complete before the message is discarded. In such a case, the DISCARDMSG service returns to the caller with the discard of the message left pending.

You can discard incomplete message-out requests before the message completes. For example, you can discard a message queued by XCF due to a lack of signalling resources or a message whose XCF-managed response collection is not complete. Such a discard has the effect of cancelling the message-out request. For a broadcast to multiple targets, any remaining unsent messages are not sent and collected responses, if any, are discarded. The system also discards any response that subsequently arrives for the cancelled message. Notification, if requested, does not occur for message-out requests that have been cancelled by this method.

Use the IXCMSGC REQUEST=COMPLETION service to force a message-out request to be stopped in a way that preserves the request for further processing by the member. For example, if the IXCMSGO invocation had specified notify exit processing, the notify exit would be driven to process the message-out request.

REQUEST=QUERYMSG

Return information appropriate for the requested type of query in the storage area specified by ANSAREA. The DATATYPE parameter specifies the type of information requested. The contents of the answer area are mapped by IXCYMQAA. Use the length and offsets provided in the IXCYMQAA records to ensure compatibility with additional data provided in the future.

REQUEST=COMPLETION

Complete the message identified by the TOKEN parameter. You can request COMPLETION for a message-out request that XCF has accepted for delivery, but does not yet consider complete.

A message is considered complete in the following situations:

- A message-out request is considered complete if it times-out or if the REQUEST=COMPLETION service is used to force its completion.
- A message with response is considered complete when the expected response(s) arrive.
- A message without response is considered complete as soon as XCF has initiated the send of the message.
- A message broadcast to multiple targets is considered complete when XCF has initiated the send of the message to every valid target member.

Once completed, processing for the message continues just as if it completed without direct intervention. XCF no longer attempts to send the message to any intended target. XCF discards any responses to the message that subsequently arrive. If notification was to be provided upon completion of the message, the message notify user routine receives control.

Use the REQUEST=DISCARDMSG service to force a message-out request to be stopped in a way that makes the message unavailable for further processing. For example, to prevent the notify exit from getting control even though it was requested on the IXCMSGO invocation.

REQUEST=CALLEXIT

Call a user routine to process the message identified by the TOKEN parameter. The message can be a saved message or a completed message-out request. The type of user routine must be appropriate for the message.

- Use a message user routine to process messages saved by a message user routine and responses saved by a message notify user routine.
- Use a message notify user routine to process completed message-out requests and saved message and response entities.

A CALLEXIT request first passes control to a system service routine that sets up the appropriate environment, collects information pertinent to the message, and calls the specified user routine. The user routine receives control synchronously — under the same unit of work as the invoker of the CALLEXIT service. After performing whatever processing is required, the user routine returns to the system service routine, which then releases resources as necessary, restores the environment, and returns control to the invoker of the CALLEXIT service.

Only one user routine is allowed to process a particular message at any one time. The system rejects the CALLEXIT request if a user routine is already processing the message.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) to contain the return code.

,RETMSGTOKEN=*retmsgtoken*

Use this output parameter to contain a token that the REQUEST=SAVEMSG service returns to identify a saved message.

Pass this token to the DISCARDMSG or CALLEXIT services of IXCMSGC. You can also obtain a token for the message to be passed to these services by invoking the QUERYMSG service of IXCMSGC.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character storage area to contain the message token.

The storage area must be in the caller's primary address space, in an address space or data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL), or in a common area data space.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the name of a fullword field (or register 2-12 containing the address of the field) to contain the reason code.

,SOURCE=ANY
,SOURCE=*source*

Use this input parameter to specify a member token. The REQUEST=QUERY service will collect only the incoming messages sent by the specified member.

To Code: Specify the name of a 64-bit field (or register 2-12 containing the address of the field) to contain the member token.

,STATUS=SAVED
,STATUS=COMPLETE
,STATUS=INCOMPLETE

Use this input parameter to specify the status of the message-out requests for which data is to be collected with REQUEST=QUERYMSG.

STATUS=SAVED

Collect information about messages that were saved through use of the REQUEST=SAVEMSG service. These messages are the message and response entities saved by the message notify user routine.

STATUS=COMPLETE

Collect information about messages that are completed. A message is considered complete in the following situations:

- A message-out request is considered complete if it times-out or if the REQUEST=COMPLETION service is used to force its completion.
- A message with response is considered complete when the expected response(s) arrive.
- A message without response is considered complete as soon as XCF has initiated the send of the message.
- A message broadcast to multiple targets is considered complete when XCF has initiated the send of the message to every valid target member.

STATUS=INCOMPLETE

Collect information about messages that have not yet completed. For example, the message might have been queued due to an XCF signalling buffer shortage, or in the case of a response-required signal, a response to the message might not have arrived and the message has either not yet timed-out or been forced to completion with IXCMSGC REQUEST=COMPLETION.

,TOKEN=*token*

Use this input parameter to identify the message to which the IXCMSGC request refers. The token used to identify a message can vary. Two different tokens might identify the same message. Do not expect to compare tokens to determine if they identify the same message. For example, a token presented to a user routine and a token obtained by other means (such as with the RETMSGOTOKEN parameter of IXCMSGO or the token obtained from the IXCMSGC SAVEMSG or the IXCMSGC QUERYMSG service) could be different and yet represent the same message. The tokens obtained by the other means described above will be the same for the same message, at least for a particular member.

A description of how to obtain a token suitable for each of the IXCMSGC services follows:

- For a SAVEMSG request, the token identifies the message to be saved. The system invalidates this token once the message is saved.

  - Obtain the token when running in a message user routine from the message token (MEPLMSGITOKEN) passed in the message exit parameter list, mapped by IXCYMEPL.
  - Obtain the token when running in a message notify user routine from the message token (MNPLMSGOTOKEN) passed in the message notification parameter list, mapped by IXCYMNPL. This token identifies the message and its associated responses (if any) as a single entity. For an individual response, obtain the token from the message token (MNPLTRMSGITOKEN) passed in the MNPL.
- For a DISCARDMSG request, the token identifies the message to be discarded.
  - Obtain the token for an incoming message from one of these sources:
    - The SAVEMSG service with the RETMSGTOKEN parameter
    - The QUERYMSG service with the DATATYPE=MSGIN
    - The MEPL (MEPLMSGITOKEN) or the MNPL (MNPLTRMSGITOKEN) if the user routine has not finished processing the message. Note that the tokens from these parameter lists are invalidated if the user routine saves the message, discards the message, receives all the message data, or gives up control. The MNPLTRMSGITOKEN also is invalidated if the message notify user routine finishes processing the message/response entity. Note also that the token is valid only under the user routine unit of work.
  - Obtain the token for a message/response entity from one of these sources:
    - The IXCMSGO service with the RETMSGOTOKEN parameter
    - The SAVEMSG service with the RETMSGTOKEN parameter
    - The QUERYMSG service with DATATYPE=MSGOUT
    - The MNPL (MNPLMSGOTOKEN) if the message notify user routine has not finished processing the message/response entity. Note that this token is invalidated if the user routine saves the message, discards the message, or gives up control.

  Each of the tokens described is invalidated if any other unit of work discards the message.
- For a QUERYMSG request, the token identifies the message for which the system is to collect DETAIL information.
  - Obtain the token to identify the message from one of these sources:
    - The SAVEMSG service with the RETMSGTOKEN parameter
    - The IXCMSGO service with the RETMSGOTOKEN parameter
    - The IXCMSGC QUERYMSG service with DATATYPE=MSGIN or MSGOUT.
- For a COMPLETION request, the token identifies the message to be completed.
  - Obtain the token to identify the message from one of these sources:
    - The IXCMSGO service with the RETMSGOTOKEN parameter
    - The IXCMSGC QUERYMSG service with DATATYPE=MSGOUT and STATUS=INCOMPLETE.
- For a CALLEXIT request, the token identifies the message to be processed by the user routine.
  - Obtain the token for a message user routine from one of these sources:
    - The IXCMSGC QUERYMSG service with DATATYPE=MSGIN
    - The IXCMSGC SAVEMSG service with RETMSGTOKEN parameter when the message was saved by a message user routine or a response was saved by a message notify user routine.
  - Obtain the token for a message notify user routine from one of these sources:
    - The IXCMSGO service with the RETMSGOTOKEN parameter
    - The IXCMSGC QUERYMSG service with DATATYPE=MSGOUT and STATUS=SAVED or COMPLETE
    - The IXCMSGC SAVEMSG service with the RETMSGTOKEN parameter when the message/response entity was saved by a message notify user routine.

,TYPE=FORCE

Use this input parameter to specify that the message is to be immediately considered as complete when invoking the IXCMSGC REQUEST=COMPLETION service. This type of completion is comparable to forcing a message time-out value to expire immediately.

,USERDATA=NO_CHANGE
,USERDATA=*userdata*

Use this input parameter to specify a storage area that contains user data to be associated with a saved or completed message. The system presents the user data to a user routine.

- For a saved message, the following fields are used to pass the user data:
  - The field MEPLEXUSERDATA contains the user data passed to a message user routine.
  - The field MNPLMSGOUSERDATA contains the user data passed to a message notify user routine.
- For a completed message, the field MNPLMSGOUSERDATA contains the user data passed to a message notify user routine. Note that the current user data associated with the message is replaced by this user data only if the invocation of the IXCMSGC COMPLETION service causes the message to be considered complete.

To Code: Specify the RS-name or address (using a register from 2 to 12) of the 8-character field that contains the user data to be associated with the message.

ABEND Codes

An abend code that an issuer of IXCMSGC might receive is listed below. For detailed abend code information, see *z/OS MVS System Codes*.

X'00C'
X'073'

Return and Reason Codes

When control returns from IXCMSGC:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable. Note that bits 0-15 of the reason code might contain component- diagnostic data for use by IBM service personnel. XCF provides the IXCMSGRSNCODEMASK constant to mask off the component-diagnostic data.

The IXCMSGC macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

00	IXCMSGCRCSUCCESSFUL
04	IXCMSGCRCWARNING
08	IXCMSGCRCINVALIDPARMS
0C	IXCMSGCRCENVIRONMENTALERROR
10	IXCMSGCRCSYSTEMERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 10. Return and Reason Codes for the IXCMSGC Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** IXCMSGC completed successfully. **Action:** None.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0004	**Equate Symbol:** IXCMSGCRSNANSAREATOOSMALL **Meaning:** The IXCMSGC QUERYMSG request completed successfully. The area specified by ANSAREA was large enough to contain the header information (MQAHEADER), but was not large enough to contain all the data that was requested. The MQAHDRTLEN field indicates the total length of the output answer area that would have been needed to contain all the requested information. It is possible that only the MQAHEADER was provided, in which case the MQAHDR#ENTRIES field would contain zero. **Action:** Retry the request with an answer area whose length is greater than or equal to the number of bytes indicated by MQAHDRTLEN. Note that the amount of data to be returned can change dynamically so that the length indicated by MQAHDRTLEN might be too small for all the data when you try the request again.
4	xxxx0008	**Equate Symbol:** IXCMSGCRSNMSGALREADYCOMPLETE **Meaning:** The message has already completed. **Action:** None. A message COMPLETION was requested for a message that was already completed.
4	xxxx0018	**Equate Symbol:** IXCMSGCRSNMSGDISCARDPENDING **Meaning:** A message-discard is pending. Some work unit is currently processing the message. (For example, the message notify user routine might be processing the message.) The system will delete the message as soon as the work unit is finished with the message. **Action:** None. The message is not available. The application might want to initiate actions to shut down the user routine. For IXCMSGO MSGACCESS=ASYNC requests, the message-out storage areas that were to be preserved until the message completed can now be freed.
8	xxxx0004	**Equate Symbol:** IXCMSGCRSNMEMBERNOTACTIVE **Meaning:** The member token does not identify an active member associated with the primary address space that was current when IXCMSGC was invoked. **Action:** Reissue the request with a correct member token or from the correct address space.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0016	**Equate Symbol:** IXCMSGCRSNINAPPROPEXITROUTINENAME **Meaning:** The user routine type specified was inappropriate for this request. • Messages saved by a message user routine and responses saved by a message notify user routine must be processed by a message user routine. Use the MSGEXIT parameter to specify the user routine. • A completed message-out request, or a saved message-out response entity must be processed by a message notify user routine. Use the NOTIFYEXIT parameter to specify the user routine. **Action:** The type of user routine specified for a CALLEXIT request must be appropriate for the type of message to be processed. Retry the request with the correct user routine.
8	xxxx0040	**Equate Symbol:** IXCMSGCRSNRESERVEDFIELDNOTNULL **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of z/OS on which your program is running.
8	xxxx0100	**Equate Symbol:** IXCMSGCRSNBADPLISTALET **Meaning:** Program error. Your program is not running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is not zero, associated with a valid public entry on the DU-AL, or for a common area data space. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCMSGC macro. • The ALET for the parameter list is a valid public entry on the DU-AL, is zero (primary address space ALET), or for a common area data spece.
8	xxxx0104	**Equate Symbol:** IXCMSGCRSNBADPLISTVERSION **Meaning:** The parameter list is not valid. Either the version number in the parameter list is not valid, or the release level of z/OS on which the caller is running does not support this version of the message control service. **Action:** Retry the request with the correct version of the parameter list.
8	xxxx0108	**Equate Symbol:** IXCMSGCRSNBADPLISTFUNCCODE **Meaning:** The parameter list is not valid. The function code in the parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list in storage, and that it was assembled with the correct macro library for the release of z/OS on which it is running.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx010C	**Equate Symbol:** IXCMSGCRSNBADPLISTADDRESS **Meaning:** The parameter list is not accessible. Its storage is not addressable. **Action:** Make sure the parameter list is accessible to XCF and retry the request.
8	xxxx011C	**Equate Symbol:** IXCMSGCRSNNOTENABLED **Meaning:** The caller is not enabled. **Action:** Correct your program so that it does not issue IXCMSGC while it is disabled.
8	xxxx012C	**Equate Symbol:** IXCMSGCRSNLOCKSHELD **Meaning:** The caller is holding a lock. **Action:** Correct your program so that it does not issue IXCMSGC while holding any locks.
8	xxxx013C	**Equate Symbol:** IXCMSGCRSNANSAREASMALLERTHANHEADER **Meaning:** The area specified by ANSAREA is too small. The answer area must be at least as long as the header record (MQAHEADER). **Action:** Retry the request with a larger answer area.
8	xxxx0140	**Equate Symbol:** IXCMSGCRSNANSAREABADALET **Meaning:** The area specified by ANSAREA is not accessible. The ALET of the answer area must be zero, a valid entry on the dispatchable unit access list (DU-AL), or a valid entry for a common area data space. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCMSGC macro. • The ALET for the parameter list is a valid public entry on the DU-AL, is zero (primary address space ALET), or for a common area data spece.
8	xxxx0148	**Equate Symbol:** IXCMSGCRSNANSAREABADADDRESS **Meaning:** An error occurred attempting to access the answer area. **Action:** Make sure the answer area is accessible to XCF and reissue the request.
8	xxxx0200	**Equate Symbol:** IXCMSGCRSNTOKENNOTFORSAVEMSG **Meaning:** The token specified by TOKEN is not valid for the SAVEMSG service. **Action:** Verify that the token specified is the MSGITOKEN provided to the message user routine or the MNPLMSGOTOKEN or MNPLTRMSGITOKEN provided to the message notify user routine and retry the request with the correct token. Ensure that you are issuing the SAVEMSG request under the user routine unit of work.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0204	**Equate Symbol:** IXCMSGCRSNTOKENNOTFORDISCARDMSG **Meaning:** The token specified by TOKEN is not valid for the DISCARDMSG service. **Action:** Check to see if your program inadvertently overlaid the parameter list in storage. Ensure that you are issuing the DISCARDMSG request under the user routine unit of work.
8	xxxx0208	**Equate Symbol:** IXCMSGCRSNTOKENFORCALLEXITINVALID **Meaning:** The token specified by TOKEN is not valid for the CALLEXIT service. Valid tokens for the CALLEXIT service are those returned by the IXCMSGC SAVEMSG service, the IXCMSGO service (RETMSGOTOKEN), and the IXCMSGC QUERYMSG service. **Action:** Ensure that the token is one that was returned by a successful invocation of the IXCMSGC SAVEMSG or QUERYMSG service or the RETMSGOTOKEN from the IXCMSGO service. Also, check to see if your program inadvertently overlaid the parameter list in storage. Note that message tokens presented to the message user routine and the message notify user routine are not valid for the CALLEXIT service.
8	xxxx020C	**Equate Symbol:** IXCMSGCRSNMESSAGEUNAVAILABLE **Meaning:** The message indicated by TOKEN is not available, either because the message does not exist or is not associated with the member indicated by MEMTOKEN. The message was either completely delivered, processed, or discarded. **Action:** Verify that the correct token was specified; if not, reissue the request with a new correct TOKEN. For a RETMSGOTOKEN or a saved token (RETMSGTOKEN), ensure that the MEMTOKEN identifies the member to which XCF presented the token. When a user routine saves the message, the message is no longer available to the user routine. If the message still exists, use the token returned via the RETMSGTOKEN keyword on the IXCMSGC invocation that saved the message to process the message. Another option is to use the token returned by the IXCMSGC QUERYMSG service.
8	xxxx0210	**Equate Symbol:** IXCMSGCRSNMESSAGETOKENINVALID **Meaning:** The TOKEN is not valid. **Action:** Verify the token and retry the request with the correct token. Also, check to see if your program inadvertently overlaid the parameter list in storage.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0220	**Equate Symbol:** IXCMSGCRSNTOKENNOTFORFORCECOMPLETION **Meaning:** The message TOKEN is not valid for Force Completion. The message token must be a token that was returned either by the IXCMSGO service with the RETMSGOTOKEN keyword or by the IXCMSGC QUERYMSG DATATYPE=MSGOUT STATUS=INCOMPLETE service. **Action:** Verify the token and retry the request with the correct token. Also, check to see if your program inadvertently overlaid the parameter list in storage.
8	xxxx0308	**Equate Symbol:** IXCMSGCRSNBADRETMSGTOKENALET **Meaning:** The ALET that qualifies the address of the RETMSGTOKEN is not zero, nor a valid entry on the dispatchable unit access list (DU-AL), nor a valid entry for a common area data space. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCMSGC macro. • The ALET for the parameter list is a valid public entry on the DU-AL, is zero (primary address space ALET), or for a common area data spece.
8	xxxx0309	**Equate Symbol:** IXCMSGCRSNBADRETMSGTOKENADDRESS **Meaning:** The RETMSGTOKEN was not accessible. The message control service was not able to store a message token in the storage area specified by RETMSGTOKEN. **Action:** Verify the storage area is accessible to XCF and retry the request.
8	xxxx030A	**Equate Symbol:** IXCMSGCRSNBADEXITFORCALLEXIT **Meaning:** For a CALLEXIT request, XCF attempted to call the user routine but the message or message notify user routine abended. The user routine address might not be valid or the user routine might have done some processing. The specified message might have been processed by the user routine before it abended. As such, the token might or might not specify a currently valid message. **Action:** If necessary, perform any recovery action and resource cleanup for the user routine. Verify the user routine address and retry the CALLEXIT request.
8	xxxx030E	**Equate Symbol:** IXCMSGCRSNTASKMODECALLEXITWITHFRR **Meaning:** For a CALLEXIT request that was made in task mode, the caller had an FRR established. **Action:** Correct your program so that it does not issue an IXCMSGC CALLEXIT request with FRRs established while in task mode.

Table 10. Return and Reason Codes for the IXCMSGC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0C14	**Equate Symbol:** IXCMSGCRSNTOKENNOTFORQUERYMSG **Meaning:** Environmental error. TOKEN not valid for IXCMSGC QUERYMSG service. **Action:** Retry the request with the correct token.
C	xxxx0C04	**Equate Symbol:** IXCMSGCRSNMSGNOTAVAILOTHEREXIT **Meaning:** The message is not available because another exit routine is currently processing the message. **Action:** Retry your request at a later time.
C	xxxx0C08	**Equate Symbol:** IXCMSGCRSNNOUSERMSGSPACEAVAIL **Meaning:** No user message space is available. All the message storage space managed by XCF on behalf of the member are full. **Action:** Use the message control DISCARDMSG service to discard one or more messages in order to make more storage available.
C	xxxx0C0C	**Equate Symbol:** IXCMSGCRSNDUALCANNOTBEEXPANDED **Meaning:** The system is unable to process a CALLEXIT, QUERYMSG, or COMPLETION request. A STOKEN that is required to be added to the current DU-AL (dispatchable unit access list) could not be added because the DU-AL was either full or not expandable. **Action:** Retry the request at a later time or remove an entry from the DU-AL and retry the request.
C	xxxx0C10	**Equate Symbol:** IXCMSGCRSNNOWORKINGSTORAGE **Meaning:** Environmental error. An IXCMSGC request could not be performed because XCF could not obtain enough working storage in the caller’s address space to process the request. **Action:** Retry your request at a later time.
C	xxxx0C18	**Equate Symbol:** IXCMSGCRSNMSGPENDING **Meaning:** Environmental error. The CALLEXIT service is not valid for a message-out request that is not yet completed. **Action:** Reissue the CALLEXIT request after the message is completed. Use the IXCMSGC COMPLETION service to force the message to be considered complete, if necessary.
10	None	**Meaning:** System failure. XCF processing failure. **Action:** Save the return and reason code information and contact the IBM Support Center.

IXCMSGI — Receive a Message from Another Member in the XCF Group

Description

The IXCMSGI macro (message-in service) allows a member of an XCF (cross-system coupling facility) group to receive a message or response from another member in its XCF group. IXCMSGI, and its companion services, IXCMSGO (message-out service) and IXCMSGC (message control service), comprise the XCF signalling services. To receive messages and responses through XCF signalling services, you must write either a message user routine or a message notify user routine that invokes the IXCMSGI macro. When an XCF member sends a message or response by issuing the IXCMSGO macro, XCF schedules either the message user routine or the message notify user routine belonging to the member receiving the message.

Messages can vary in length up to 128M bytes. A “large message” is defined to be one that is greater than 62464 (61K) bytes. XCF handles large messages somewhat differently from messages of less than 61K bytes, and also imposes the following restrictions when sending large messages:

- Both the sending and the target **systems** must be running OS/390 Release 8 or higher. (This does not necessarily imply that all **members** residing on those systems support large message delivery.)
- Both the sending and the target **members** must have specified when they joined the XCF group that they supported large message delivery. (Each must have specified GT61KMSG=YES on the IXCJOIN invocation.)

Members of an XCF group can receive message data using either a single buffer or multiple buffers.

For More Information

Be sure to read the IXCMSGI guidance information in *z/OS MVS Programming: Sysplex Services Guide*. The information presented here assumes you have done so. IXCMSGI guidance information includes:
- Illustrations showing the message data element formats specified by the various IXCMSGI parameters.
- Information on designing a protocol for sending and receiving multipart messages.
- Information on writing a message user routine and a message notify user routine.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	HASN=PASN, any SASN. The primary address space must equal the primary address space of the caller of the IXCJOIN macro when it was issued to join the XCF group. IXCMSGI must be invoked under the work unit of the message or message notify user routine.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space

Programming Requirements

- If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before issuing IXCMSGI to generate code for AR mode.
- The IXCYMEPL mapping macro provides the format of the message exit parameter list (MEPL) that XCF passes to the message user routine. Include that mapping macro in the code for the message user routine. For more information about the MEPL, see *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.
- If you specify the MSGBUF parameter, the buffer it designates must be large enough to receive the entire message. If the buffer is too small you will either receive a program check or overwrite storage.
- For MULTIPART=YES, the invoker of IXCMSGI is responsible for maintaining the integrity of the element structure until the message-in service returns. If the elements or their structure change while being processed by the message-in service, the message data actually received may be corrupted.
- All message data elements (but not buffers whose addresses are located in the message data elements) must reside in the same address space or data space so they can be accessed using the same ALET.

Restrictions

- The IXCMSGI macro can be issued within a message user routine or message notify user routine that XCF schedules on behalf of the receiving member or to which XCF gives control on behalf of the IXCMSGC CALLEXIT service.
- The sending and receiving members must be active members of the same XCF group.

Input Register Information

Before issuing the IXCMSGI macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0	Reason code, if GPR 15 contains a non-zero return code that has applicable reason codes.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Understanding IXCMSGI Version Support

The IXCMSGI macro supports versions in the range of 0 - 2.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXCMSGI macro.
- The following keywords and functions are supported by version 1 and subsequent versions of the IXCMSGI macro.

ELEMENT
ELEMFORM
ENDOFQUEUE
MSGSTGKEY
NEXTOFF
NEXTPTROFF
PARTALET
PARTALETOFF
PARTALETTBL
PARTLEN
PARTLENOFF
PARTLENTBL
PARTOFF
PARTPTROFF
#MSGPARTS

- The following keyword is supported by version 2 and subsequent versions of the IXCMSGI macro.

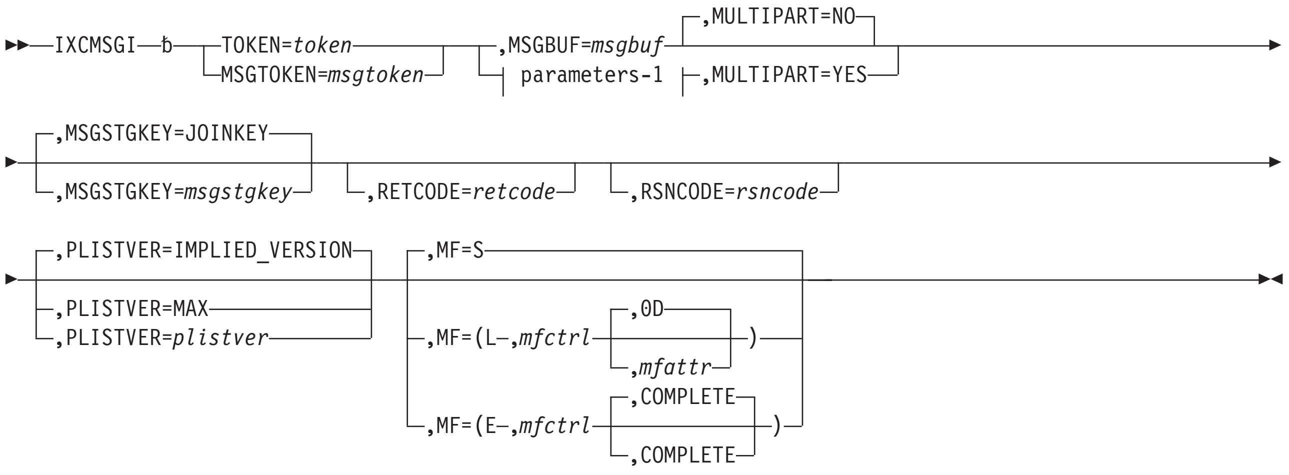
TOKEN

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCMSGI macro is as follows:

main diagram



parameters-1

```
   ┌─,#MSGPARTS=AS_NEEDED─┐
►►─┤                      ├─,ELEMENT=element──────────────────────────────────────────►
   └─,#MSGPARTS=#msgparts─┘

►─,ELEMFORM=─┬─TABLE──,NEXTOFF=nextoff───────────────────────────────────────────┬───►
             │                                   ┌─,ENDOFQUEUE=ZERO────────┐      │
             └─QUEUE──,NEXTPTROFF=nextptroff─────┼─────────────────────────┼──────┘
                                                 └─,ENDOFQUEUE=endofqueue──┘

►──┬─,PARTOFF=partoff─────────────────────────────────────────┬──┬─,PARTLEN=partlen─────────┬──────►◄
   │                          ┌─,PARTALET=ZERO──────────────┐  │  ├─,PARTLENOFF=partlenoff───┤
   └─,PARTPTROFF=partptroff───┼─────────────────────────────┼──┘  └─,PARTLENTBL=partlentbl───┘
                              ├─,PARTALET=partalet──────────┤
                              ├─,PARTALETOFF=partaletoff────┤
                              └─,PARTALETTBL=partalettbl────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ELEMENT=*element*

Use this input parameter to specify the first element of the table or queue of message data elements. Message data elements contain either buffers or pointers to buffers that are to receive parts of the message.

Specifying the PARTOFF parameter indicates that each element contains a buffer. Specifying the PARTPTROFF parameter indicates that each element contains a pointer to a buffer.

Elements must all reside in the same space, either an address space or a data space. Elements can be in the caller's primary address space, addressable through a public entry on the DU-AL, or in a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the first element.

,ELEMFORM=TABLE
,ELEMFORM=QUEUE

Use ELEMFORM=TABLE to indicate that the message data elements are organized as a table.

Use ELEMFORM=QUEUE to indicate that the message data elements are organized as a queue.

,ENDOFQUEUE=<u>ZERO</u>
,ENDOFQUEUE=*endofqueue*

Use this input parameter to specify the address that marks the end of the queue. When the pointer to the next element contains this address, queue processing ends. You can use either ENDOFQUEUE or #MSGPARTS to limit the amount of message data elements that are processed. A partial delivery occurs if the number of message data elements is insufficient to contain all the data. To receive more of the data, continue to reissue IXCMSGI until all data has been delivered.

Note: The queue must have at least one element.

If you omit ENDOFQUEUE, the default value for the end-of-queue address is 0 (you can code this explicitly by specifying ENDOFQUEUE=ZERO).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the end-of-queue address.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MSGBUF=*msgbuf*

Use this output parameter to specify the buffer to receive the incoming message. The size of the buffer must be greater than or equal to the value of MEPLMLEN in IXCYMEPL (if processing in a message user routine) or MNPLTRRESPMLEN in IXCYMNPL (if processing in a message notify user routine). The storage key of the buffer specified by MSGBUF must match the storage key specified with the MSGSTGKEY parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a storage area to receive the message.

MSGTOKEN=*msgtoken*

Use the TOKEN keyword instead of the MSGTOKEN keyword. MSGTOKEN is not supported for IXCMSGI requests invoked from a message notify user routine. MSGTOKEN is provided for compatibility with:

- Message user routines that are running on systems with releases prior to OS/390 Release 3.
- Older applications that have been assembled with pre-OS/390 Release 2 code but that are running on an OS/390 Release 3 or later system.

Otherwise, use this input parameter to specify the message token that your message user routine received from XCF in the MEPLMTOK field of the MEPL (mapped by IXCYMEPL).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-bit field that contains the message token.

,MSGSTGKEY=JOINKEY
,MSGSTGKEY=*msgstgkey*

Use this input parameter to specify the storage key to be used by XCF when storing the message into each receiving buffer. The storage key of each receiving buffer must match the storage key specified by MSGSTGKEY.

If you omit the MSGSTGKEY parameter, or if you specify MSGSTGKEY=JOINKEY, XCF uses as the storage key the value of the PSW key at the time you joined the XCF group (when IXCJOIN was issued).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field formatted as **kkkkxxxx**, where the high-order four bits contain the storage key. The low-order four bits are ignored.

,MULTIPART=NO
,MULTIPART=YES

Use MULTIPART=NO to indicate that the message is to be received into a single buffer.

Use MULTIPART=YES to indicate that the message is to be received into one or more buffers.

,NEXTOFF=*nextoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element to obtain the address of the next element. This value equals the size in bytes of an individual message data element. It is used when the message data elements are in table form.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes to be added.

,NEXTPTROFF=*nextptroff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the current element the fullword pointer to the next element. This value is used when the message data elements are in queue form.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTALET=ZERO
,PARTALET=*partalet*

Use this input parameter to specify a single ALET to be used to qualify every buffer address. The ALET must be zero, a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

If you omit PARTALET, PARTALETOFF, and PARTALETTBL, the default is PARTALET with an ALET of 0 (if you would like to code this explicitly, specify PARTALET=ZERO).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the ALET.

,PARTALETOFF=*partaletoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword field that contains the ALET to be used to qualify the buffer address associated with that element. The ALET must be zero, a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTALETTBL=*partalettbl*

Use this input parameter to specify a table in which each entry is a fullword containing the ALET to qualify a buffer address in the table or queue of message data elements. For instance, the 3rd entry in PARTALETTBL must contain the ALET to qualify the buffer address in the 3rd message data element. *partalettbl* must begin on a fullword boundary. Each ALET must be zero,a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of *partalettbl*.

,PARTLEN=*partlen*

Use this input parameter to specify the length in bytes of each buffer when all the buffers are the same length. All buffers must be at least as long as PARTLEN.

To receive the entire message in a message user routine, the total amount of storage for the message (the value of PARTLEN multiplied by the number of elements) must be greater than or equal to MEPLMLEN. For a message notify user routine to receive the entire response, the total amount of storage must be greater than or equal to MNPLTRRESPMLEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the length in bytes of each buffer.

,PARTLENOFF=*partlenoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword field that contains the length in bytes of the buffer associated with that element.

The value of the field indicated by PARTLENOFF should contain a value at least as long (but may be less than or equal to the length of the buffer) as the part of the message you want to put in this buffer. If you specify a length of zero for the message part, XCF does not use that buffer.

To receive the entire message, the total amount of storage for the message (the sum of the buffer lengths) must be greater than or equal to MEPLMLEN. For a message notify user routine to receive an entire response, the total amount of storage must be greater than or equal to MNPLTRRESPMLEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTLENTBL=*partlentbl*

Use this input parameter to specify a table in which each entry is a fullword containing the length of a corresponding buffer. For instance, the 3rd entry in PARTLENTBL contains the length of the buffer whose address is associated with the 3rd message data element in the table or queue. If you specify the length of a buffer as zero, XCF does not use that buffer.

The table specified by PARTLENTBL must begin on a fullword boundary.

To receive the entire message, the total amount of storage for the message (the sum of the buffer lengths) must be greater than or equal to MEPLMLEN. For a message notify user routine to receive the entire response, to total amount of storage must be greater than or equal to MNPLTRRESPMLEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of *partlentbl*.

,PARTOFF=*partoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to obtain the address within the element of the start of the buffer associated with it.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTPTROFF=*partptroff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword pointer to the buffer associated with it.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXCMSGI Version Support” on page 116 for a description of the options available with PLISTVER.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

TOKEN=*token*

Use this input parameter to specify the location of the 16-character field that contains a token that identifies the message whose message data is to be delivered.

The token identifies one of the following:

- A message presented to a message user routine. On entry to the routine, R1 contains the address of the message exit parameter list (MEPL) that contains the token (MEPLMSGITOKEN).
- A response message presented to a message notify user routine. On entry to the routine, R1 contains the address of the message notification parameter list (MNPL) that contains the token (MNPLTRMSGITOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field containing the token.

,#MSGPARTS=AS_NEEDED
,#MSGPARTS=*msgparts*

Use this input parameter to specify the number of buffers (1 or more) to be used to receive the message. The number of buffers specified determines whether an entire message is received or only part of the data. You can use #MSGPARTS to limit the amount of message data you receive. To receive some but not all of the message or response, specify #MSGPARTS to be less than the number of buffers that would be needed to receive all the data. To receive an entire message or response, you can omit the #MSGPARTS parameter, specify #MSGPARTS=AS_NEEDED, or specify a number of buffers that is sufficient to contain all the data.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of buffers.

ABEND Codes

None.

Return and Reason Codes

When the IXCMSGI macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK

4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 11. Return and Reason Codes for the IXCMSGI Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
00	None.	**Meaning:** IXCMSGI completed successfully; the message data has been received as specified. **Action:** None.
04	224	**Equate Symbol:** IXCMSGIRSNSTILLMOREDATA **Meaning:** The total amount of buffer storage was insufficient. The buffers contain as much of the message as could fit. There is more message data to receive. This return code and reason code are only returned if you specified MULTIPART=YES. **Action:** Continue to reissue the IXCMSGI macro to receive the remainder of the message. Alternately, you could save or discard the remainder of the message.
08	04	**Equate Symbol:** IXCMSGIRSNMSGBUFBADSTG **Meaning:** Program error. XCF could not access the buffer specified by MSGBUF. **Action:** Check for errors such as the following; • The buffer address is incorrect. • The buffer was previously freed. • If your program is running in AR mode, check for the following types of errors: – The message buffer ALET is incorrect. – You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro.
08	08	**Equate Symbol:** IXCMSGIRSNALREADYDELIVERED **Meaning:** Program error. XCF already delivered the message, and it is no longer available. **Action:** If this result was acceptable, no action might be necessary. Otherwise, check for errors such as the following: • You specified an incorrect message token. • You tried to receive the same message more than once. • Your user routine is not reentrant.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	09	**Equate Symbol:** IXCMSGIRSNMEMBERNOTACTIVE **Meaning:** Program error. The message is no longer available because the member is no longer active. **Action:** If this result was acceptable, no action might be necessary. Otherwise: • Check to see if you specified an incorrect message token. • Have your user routine give up control.
08	0C	**Equate Symbol:** IXCMSGIRSNMSGBUFBADALET **Meaning:** Program error. The ALET that qualifies the address of the buffer specified by MSGBUF must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro. • You specified an incorrect buffer area ALET.
08	40	**Equate Symbol:** IXCMSGIRSNPLISTRSVDNOTVALID **Meaning:** Program error. A reserved field in the control parameter list was not zero. **Action:** Check for errors such as the following: • Your program overlaid the control parameter list storage. • Your program was assembled with the wrong macro library for the release of z/OS on which your program is running.
08	44	**Equate Symbol:** IXCMSGIRSNMSGTOKENNOTVALID **Meaning:** Program error. The message token was not valid. **Action:** Check for errors such as the following: • The message token you specified was not the one contained in the MEPLMTOK field of the MEPL, the MEPLMSGITOKEN field of the MEPL, or the MNPLTRMSGITOKEN field of the MNPL. • Your program overlaid the control parameter list storage. • Your program is not reentrant.
08	45	**Equate Symbol:** IXCMSGIRSNUSETOKENKEYWORD **Meaning:** Program error. MSGTOKEN is not valid. Use the TOKEN keyword. **Action:** Correct your program to use the TOKEN keyword, which is required when invoking a message user routine through a message notify user routine.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	100	**Equate Symbol:** IXCMSGIRSNPLISTBADALET **Meaning:** Program error. The ALET that qualified the address of the control parameter list was not zero. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro. • The ALET for the parameter list storage was not zero (primary address space ALET). **Note:** The ALET for the control parameter list must be zero.
08	104	**Equate Symbol:** IXCMSGIRSNPLISTVERSIONNOTVALID **Meaning:** Program error. The version number in the control parameter list was not valid. **Action:** Check for errors such as the following: • Your program overlaid the control parameter list storage. • Your program was assembled with the wrong macro library for the release of z/OS on which your program is running.
08	10C	**Equate Symbol:** IXCMSGIRSNPLISTBADSTG **Meaning:** Program error. The control parameter list could not be accessed. **Action:** Check for errors such as the following: • You specified an incorrect address for the control parameter list. • The control parameter list storage area had been previously freed.
08	20C	**Equate Symbol:** IXCMSGIRSNMSGBUFSTGKEYMISMATCH **Meaning:** Program error. The service could not store the message data into the buffer area specified by MSGBUF using the storage key specified by MSGSTGKEY or the default storage key (if MSGSTGKEY was not specified, or if MSGSTGKEY=JOINKEY was specified). **Action:** Check for errors such as the following: • The buffer area was allocated in a storage key that did not match the key specified using the MSGSTGKEY parameter. • MSGSTGKEY was not specified, or MSGSTGKEY=JOINKEY was specified, and the buffer area was allocated in a storage key that did not match the key of the caller of IXCJOIN.
08	20D	**Equate Symbol:** IXCMSGIRSNMSGBUFPAGEPROTECT **Meaning:** Program error. The buffer specified by MSGBUF was page-protected (read-only) so the message data could not be placed in the buffer area. **Action:** Provide IXCMSGI with buffer storage that is not page-protected.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	210	**Equate Symbol:** IXCMSGIRSNPARTPTROFFBADSTG **Meaning:** Program error. The element is not accessible. The address within an element at the offset specified by PARTPTROFF is not accessible. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the invalid address. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the invalid address. **Action:** Check for errors such as the following: • You specified an incorrect element address. • Your program overlaid the element address. • The element had been previously freed. • You specified an incorrect number of buffers to receive the message. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value. • If your program is running in AR mode, check for the following: – You specified the wrong ALET for the buffer. – You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro.
08	212	**Equate Symbol:** IXCMSGIRSNELEMENTBADALET **Meaning:** Program error. The ALET that qualifies the address of an ELEMENT must be zero, a public entry on your DU-AL or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro. • You specified the wrong ALET for the element.
08	213	**Equate Symbol:** IXCMSGIRSNNEXTPTROFFBADSTG **Meaning:** Program error. The next element pointer at offset NEXTPTROFF within an ELEMENT is not accessible. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the ELEMENT containing the incorrect NEXTPTROFF field. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the incorrect address. **Action:** Check for errors such as the following: • You specified an incorrect element address. • Your program overlaid the element address. • You specified an incorrect number of buffers to receive the message. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	214	**Equate Symbol:** IXCMSGIRSN#MSGPARTSZERO **Meaning:** Program error. The value specified by the #MSGPARTS parameter was zero but must be greater than zero. **Action:** Specify a value greater than zero for #MSGPARTS or omit the #MSGPARTS parameter.
08	215	**Equate Symbol:** IXCMSGIRSNTOOMANYZEROLENPARTS **Meaning:** Program error. The message data element table or queue is assumed to be damaged. You omitted the #MSGPARTS parameter, and XCF processed more than 65,536 consecutive elements with buffers of length zero. **Action:** Check for errors such as the following: • You specified PARTLEN with a value of zero. • You specified PARTLENTBL but the address of the table was not correct. • You specified PARTLENOFF with an incorrect value. • You specified an incorrect end-of-queue value. Ensure that the elements have not been corrupted. Specify the #MSGPARTS parameter when invoking IXCMSGI.
08	218	**Equate Symbol:** IXCMSGIRSNPARTPTROFF@BADSTG **Meaning:** Program error. A buffer whose address was at the offset specified by PARTPTROFF could not be accessed. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the incorrect address. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the incorrect address. **Action:** Check for errors such as the following: • The buffer address is incorrect. • The buffer was previously freed. • You specified PARTPTROFF with an incorrect value. • If your program is running in AR mode, check for the following types or errors: – You specified the wrong ALET for the buffer. – You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	219	**Equate Symbol:** IXCMSGIRSNPARTOFFBADSTG **Meaning:** Program error. A buffer located at the offset specified by PARTOFF could not be accessed. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the buffer that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer that could not be accessed. **Action:** Check for errors such as the following: • The storage containing the element has been freed. • The PARTOFF value is incorrect. • The storage containing the element has been overlaid.
08	21A	**Equate Symbol:** IXCMSGIRSNPARTPTROFF@PAGEPROTECT **Meaning:** Program error. A buffer whose address was at the offset specified by PARTPTROFF could not be used because it was page-protected (read-only). In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the address of the page-protected buffer. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the address of the page-protected buffer. **Action:** Provide IXCMSGI with buffer storage that is not page-protected.
08	21B	**Equate Symbol:** IXCMSGIRSNPARTOFFPAGEPROTECT **Meaning:** Program error. A buffer located at the offset specified by PARTOFF could not be used because it was page-protected (read-only). In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the page-protected buffer. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the page-protected buffer. **Action:** Provide IXCMSGI with buffer storage that is not page-protected.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	21C	**Equate Symbol:** IXCMSGIRSNPARTPTROFF@KEYMISMATCH **Meaning:** Program error. XCF could not store the message data into a buffer whose address was at the offset specified by PARTPTROFF using the storage key specified by MSGSTGKEY or the default storage key (if MSGSTGKEY was not specified, or if MSGSTGKEY=JOINKEY was specified). In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the address of the buffer that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the address of the buffer that could not be accessed. **Action:** Check for errors such as the following: • The buffer was allocated in a storage key that did not match the key specified using the MSGSTGKEY parameter. • MSGSTGKEY was not specified or if MSGSTGKEY=JOINKEY was specified, and the buffer area was allocated in a storage key that did not match the key of the caller of IXCJOIN.
08	21D	**Equate Symbol:** IXCMSGIRSNPARTOFFKEYMISMATCH **Meaning:** Program error. XCF could not store the message data into a buffer area located at the offset specified by PARTOFF using the storage key specified by MSGSTGKEY or the default storage key (if MSGSTGKEY was not specified or if MSGSTGKEY=JOINKEY was specified). In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the buffer that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer that could not be accessed. **Action:** Check for errors such as the following: • The buffer area was allocated in a storage key that did not match the key specified using the MSGSTGKEY parameter. • MSGSTGKEY was not specified, or if MSGSTGKEY=JOINKEY was specified, and the buffer area was allocated in a storage key that did not match the key of the caller of IXCJOIN.
08	220	**Equate Symbol:** IXCMSGIRSNPARTLENTBLBADSTG **Meaning:** Program error. The table specified by PARTLENTBL could not be accessed. **Action:** Check for errors such as the following: • PARTLENTBL was inadvertently freed. • The element was overlaid. • You specified an incorrect address for the table specified by PARTLENTBL.
08	221	**Equate Symbol:** IXCMSGIRSNPARTLENTBLNOTWORDBDY **Meaning:** Program error. The table specified by PARTLENTBL does not begin on a fullword boundary. **Action:** Ensure that the table specified by PARTLENTBL begins on a fullword boundary.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	222	**Equate Symbol:** IXCMSGIRSNPARTLENTBLBADALET **Meaning:** Program error. The ALET that qualifies the address of PARTLENTBL must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro. • You specified the wrong ALET for PARTLENTBL.
08	223	**Equate Symbol:** IXCMSGIRSNPARTLENOFFBADSTG **Meaning:** Program error. The buffer length located at the offset specified by PARTLENOFF could not be accessed within an ELEMENT. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the buffer length that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer length that could not be accessed. **Action:** Check for following types of errors: • The storage containing the element has been freed. • The PARTLENOFF value is incorrect. • The storage containing the element has been overlaid. • You specified an incorrect number of buffers to receive the message. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.
08	230	**Equate Symbol:** IXCMSGIRSNPARTALETTBLBADSTG **Meaning:** Program error. The table specified by PARTALETTBL could not be accessed. **Action:** Check for errors such as the following: • PARTALETTBL was inadvertently freed. • PARTALETTBL was overlaid or incorrect. • You specified an incorrect address for the table specified by PARTALETTBL.
08	231	**Equate Symbol:** IXCMSGIRSNPARTALETTBLNOTWORDBDY **Meaning:** Program error. The table specified by PARTALETTBL does not begin on a fullword boundary. **Action:** Ensure that the table specified by PARTALETTBL begins on a fullword boundary.
08	232	**Equate Symbol:** IXCMSGIRSNPARTALETTBLBADALET **Meaning:** Program error. The ALET that qualifies the address of PARTALETTBL must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGI macro. • You specified the wrong ALET for PARTALETTBL.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	233	**Equate Symbol:** IXCMSGIRSNPARTALETOFFBADSTG **Meaning:** Program error. The ALET located at the offset specified by PARTALETOFF could not be accessed. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the ALET that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that could not be accessed. **Action:** Check for errors such as the following: • The storage containing the element has been freed. • The PARTALETOFF value is incorrect. • The storage containing the element has been overlaid. • You specified an incorrect number of buffers to receive the message. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.
08	234	**Equate Symbol:** IXCMSGIRSNPARTALET@BADALET **Meaning:** Program error. The ALET specified by PARTALET was not valid. The ALET must be zero, a valid public entry on your DU-AL, or an entry for a common area data space. **Action:** Specify an ALET that is zero, a public entry on your DU-AL, or an entry for a common area data space.
08	235	**Equate Symbol:** IXCMSGIRSNPARTALETTBL@BADALET **Meaning:** Program error. An ALET specified in a PARTALETTBL entry was not valid. Each ALET must be zero, a public entry on your DU-AL, or an entry for a common area data space. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the ALET that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that was not valid. **Action:** Correct the ALET.

Table 11. Return and Reason Codes for the IXCMSGI Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	236	**Equate Symbol:** IXCMSGIRSNPARTALETOFF@BADALET **Meaning:** Program error. An ALET specified at the offset designated by PARTALETOFF was not valid. Each ALET must be zero, a public entry on your DU-AL, or an entry for a common area data space. In the parameter list generated by IXCMSGI: • The field, PART#, indicates the index of the element containing the ALET that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that was not valid. **Action:** Correct the ALET. Check for errors such as the following: • You specified an incorrect number of buffers to receive the message. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.
08	010Cxxxx	**Equate Symbol:** IXCMSGIRSNPLISTNOPARTINFOBADSTG **Meaning:** Program error. A parameter specifying information on a request with MULTIPART=YES was not valid. The system was unable to store the erroneous information in the PART# and ELEMENTPTR fields in the parameter list generated by IXCMSGI because it could not access the parameter list. The low-order half word of the reason code (xxxx) indicates the reason code for the error that would have been returned if the parameter list had been accessible. **Action:** The control parameters must be in the primary address space. Make sure that the storage for the parameter list was not inadvertently freed.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return and reason code and supply it to the appropriate IBM support personnel.

Example

Operation: Receive a message that another member of the XCF group sent. The token and length for the message are accessed from the parameter list at entry to the message user-routine (using the IXCYMEPL mapping macro). Register 4 is the length of the message; the STORAGE macro obtains an area equal to that length. Register 3 contains the address of this area. TOKENMSG contains the token of the message to be received. XCF is to store the return code and reason code into the variables RETURN and REASON. Although not shown here, you must obtain storage for the dynamic area (@DATD). See “Using the Cross-System Coupling Facility” in *z/OS MVS Programming: Sysplex Services Guide*, for a complete example showing the use of IXCMSGI in a message user routine.

Note: Since more than one message user routine can be running at a time, **message user routines should be reentrant**. IBM recommends that you use the list and execute forms of IXCMSGI.

```
L     R4,MEPLMLEN                 LENGTH OF INCOMING MESSAGE
STORAGE OBTAIN,LENGTH=(R4),SP=0   GET STORAGE FOR MESSAGE
LR    R3,R1                       SAVE ADDRESS OF STORAGE
MVC   TOKENMSG(4),MEPLMTOK        MESSAGE TOKEN TO PASS TO XCF
```

```
         IXCMSGI MSGTOKEN=TOKENMSG,MSGBUF=(R3),RETCODE=RETURN,         X
               RSNCODE=REASON,MF=(E,MSGILSTD)    HAVE XCF PUT          X
                                             MESSAGE IN STORAGE        X
                                             JUST OBTAINED


* DSECT used to map the dynamic area storage
@DATD            DSECT
                 DS 0F
MSGILST2         IXCMSGI MF=(L,MSGILSTD)    LIST FORM OF IXCMSGI MACRO
RETURN           DS 1F                      RETURN CODE
REASON           DS 1F                      REASON CODE
TOKENMSG         DS CL4                     MESSAGE TOKEN TO PASS TO XCF
```

You can obtain the message token from the MEPLMTOK field in the parameter list passed to the message user routine. The message token identifies the message to be received. Do not confuse the message token with the member token. The member token identifies the member that sent the message. You can obtain the member token from the MEPLSRCE field in the parameter list.

IXCMSGO — Send a Message to Another Member in the XCF Group

Description

The IXCMSGO macro (message-out service) allows a member of an XCF (cross-system coupling facility) group to send a message to one or more members in its XCF group. IXCMSGO, and its companion services, IXCMSGI (message-in service) and IXCMSGC (message control service), comprise the XCF signalling services. The target XCF member can receive the message by issuing the IXCMSGI macro from within a message user routine it defines when it joins the XCF group. The target member can also save or discard the message by issuing the IXCMSGC macro from within the message user routine. Between pairs of members, you can specify that XCF is to deliver messages to a target member in the same order in which they were sent.

Messages can vary in length up to 134,217,278 (128M) bytes. A “large message” is defined to be one that is greater than 62464 (61K) bytes. XCF handles large messages somewhat differently from messages of less than 61K bytes, and also imposes the following restrictions when sending large messages:

- Both sending and target **systems** must be running OS/390 Release 8 or higher. (This does not necessarily imply that all **members** residing on those systems support large message delivery.)
- Both sending and target **members** must have specified when they joined their XCF group that they supported large message delivery. (Each must have specified GT61KMSG=YES on the IXCJOIN invocation.)

The following IXCMSGO error notifications occur if either of the above conditions is not met:

- If a **target system** is not at the appropriate release level, requests to send a large message to a member on that system are rejected with return code X'8', reason code X'340'. If a **source system** is not at the appropriate release level, requests to send a large message from a member on that system are rejected with return code X'8', reason code X'40'.
- If a sending **member** has not specified support for large message delivery when joining the XCF group, a request to send a large message is rejected with return code X'8', reason code X'C'. If a target **member** has not specified support for large message delivery when joining the XCF group, a request to send a large message is rejected with return code X'8', reason code X'340'.

You pass the message to the IXCMSGO service using either a single buffer or multiple buffers. If you are using a single buffer, you need only:

- Identify the message buffer using the MSGBUF parameter.
- Identify the message length using the MSGLEN parameter.

If you are sending a message that resides in multiple buffers, you have several options for specifying the location and size of the message buffers. For each message buffer, you must describe a **message data element**. Message data elements can either contain the message buffer or provide a pointer to it. Message data elements can optionally contain the length of the buffer and an ALET to qualify the buffer address (if the buffer is not in message data element).

You can pass the message data elements either organized as a queue or a table. Message buffer lengths and message buffer ALETs can also be passed separately if you do not wish to include them in the message data elements. A message buffer can be in your primary address space, in an address space accessible through your dispatchable unit access list (DU-AL), or in a common area data space. *z/OS MVS Programming: Sysplex Services Guide* provides a complete description of the different options for passing messages in multiple buffers.

The buffer storage in which the message resides is accessed by XCF when IXCMSGO is invoked. XCF can access this storage in one of two ways, as specified by the MSGACCESS keyword.

- With MSGACCESS=SYNC, XCF accesses the storage synchronously with the message-out request. Once IXCMSGO returns to the sender, the sender can dispose of the storage that contained either the message or the queue or table that defined the parts of the message.

  Note that a request to send a message longer than 61K bytes will be rejected with return code X'8', reason code X'C' when MSGACCESS=SYNC is specified.
- With MSGACCESS=ASYNC, XCF can access the storage even after IXCMSGO returns to the sender. If IXCMSGO returns with return code X'4', reason code X'410', the sender must preserve the storage containing the message text as well as the queue or table that defined the parts of the message until the message is completed. To determine when a message is complete, either
  - Wait for XCF to notify the sender via the message notify user routine.
  - Invoke the IXCMSGC service. (IXCYMQAA contains an indication as to whether the message has completed.)

  For any other return and reason codes, the sender can dispose of the storage as with MSGACCESS=SYNC.

  Note that for large message delivery, you must specify MSGACCESS=ASYNC. Also for large message delivery, you must specify a non-zero TIMEOUT value. See *z/OS MVS Programming: Sysplex Services Guide* for additional information about specifying a timeout value.

Messages you send using IXCMSGO are delivered asynchronously. A return code of zero from IXCMSGO indicates that XCF has accepted the message; it does not indicate that the message has been delivered. If it is necessary for the sender to know the message has been received, then it is up to the sender and receiver to maintain a protocol to indicate that the message has been received. Alternatively, the sender can request that the target member is to respond to the message and that XCF is to manage the response or the collection of responses, if the message is sent to more than one target member.

If you do not specify the #MSGPARTS parameter and, while looking for your message data, IXCMSGO finds more than 65536 consecutive buffers of length zero, IXCMSGO assumes an error has occurred. The message is not sent, and you receive a return code and reason code indicating the error.

For More Information

Be sure to read the IXCMSGO guidance information in *z/OS MVS Programming: Sysplex Services Guide*. The information presented here assumes you have done so. IXCMSGO guidance information includes:
- Illustrations showing the message data element formats specified by the various IXCMSGO parameters.
- Design considerations relating to the use of signalling services.
- Information on designing a protocol for sending and receiving multipart messages.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0-7
Dispatchable unit mode:	Task or SRB

Cross memory mode:	Any HASN, any SASN. Unless MSGOUTASID=ANY was specified at the time the group was joined, the primary address space must equal the primary address space of the caller of the IXCJOIN macro when it whas issued to join the XCF group; or you must be running in the master scheduler address space. If MSGOUTASID=ANY was specified at the time the group was joined, IXCMSGO can be issued from any address space.
AMODE:	31-bit
ASC mode:	Primary or access register (AR).
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCMSGO. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

For MULTIPART=YES, the invoker of IXCMSGO is responsible for maintaining the integrity of the element structure. If the elements or their structure change while being processed by the message-out service, the message data actually sent may be corrupted.

- If MSGACCESS=SYNC was specified, the caller is responsible for maintaining the integrity of the element structure until the message-out service returns.
- If MSGACCESS=ASYNC was specified, the caller is responsible for maintaining the integrity of the element structure until the message is completed.

All message data elements (but not buffers whose addresses are located in the message data elements) must reside in the same address space or data space so they can be accessed using the same ALET.

Restrictions

- The sending and receiving members must be active members of the same XCF group.
- When invoked from an address space resource manager such as the MASTER address space, some IXCMSGO functions might not be available. Any request that requires the use of one of the member data spaces that XCF manages on behalf of the member is subject to rejection. XCF attempts to perform the requested function without use of these data spaces. If a member data space is required, the request is rejected. Do not use the following keywords if your application cannot tolerate such rejections: SENDTO(GROUP), GETRESPONSE(YES), NOTIFY(YES), or TIMEOUT.

  When MSGACCESS=ASYNC is specified, if the sending member's associated task or address space or both is undergoing termination, XCF will not be able to asynchronously access the sender's data areas. (See the MEMASSOC keyword on the IXCJOIN service.) Thus, an IXCMSGO request issued from the sender's task or address space resource manager termination routine that completes with return code X'4', reason code X'410' will not be able to successfully complete the send of the message.
- When MSGACCESS=ASYNC is specified, the storage indicated by MSGBUF is subject to the following restrictions:
  - If the storage is in the caller's primary address space and this space is not the same space that was primary when the sending member joined its XCF group (that is, the joiner's address space), the caller's primary address space must be non-swappable.
  - If the storage is in a data space accessible via a public entry on the caller's DU-AL, the data space must either be owned by the joiner's address space, or be owned by a non-swappable address space, or be a common area data space.
  - If the storage is in an address space accessible via a public entry on the caller's DU-AL, that address space must either be the joiner's address space or be a non-swappable address space.

- Also when MSGACCESS=ASYNC is specified, the Dispatchable Unit Access List (DU-AL) under which the caller is running must adhere to the following restrictions. The DU-AL must never have had access to a subspace, must never have had access to more than 255 spaces of any kind at one time, and must not be full. If the Message-Out service is unable to process the request because the DU-AL is unsuitable, the IXCMSGO request is rejected with return code X'C' and an appropriate reason code.

Input Register Information

Before issuing the IXCMSGO macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code, if GPR 15 contains a non-zero return code that has applicable reason codes.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

To prevent backlogs that can degrade system performance, a target must be able to process messages as fast, or faster than the sender creates them.

For the best performance, specify the message length using the MSGLEN parameter. If you omit the MSGLEN parameter, IXCMSGO must perform processing to determine the message length.

The IXCMSGO services that incur the least overhead when sending a message are:
- SENDTO=MEMBER
- NOTIFY=NO
- MSGBUF=x
- UNORDERED
- GETRESPONSE=NO

Other IXCMSGO services require additional processing.

Understanding IXCMSGO Version Support

The IXCMSGO macro supports versions in the range of 0 - 3.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXCMSGO macro.

- The following keywords and functions are supported by version 1 and subsequent versions of the IXCMSGO macro.

ELEMENT	NEXTPTROFF	PARTLENOFF
ELEMFORM	PARTALET	PARTLENTBL
ENDOFQUEUE	PARTALETOFF	LARTOFF
MSGSTGKEY	PARTALETTBL	PARTPTROFF
NEXTOFF	PARTLEN	#MSGPARTS

- The following keywords and functions are supported by version 2 and subsequent versions of the IXCMSGO macro.

DELIVERMSG	NOTIFYBY	STREAMID
GETRESPONSE	NOTIFYEXIT	TARGETS
MEMBERS	NOTIFYIF	TIMEOUT
NEXTTARGETOFF	RESPONSEID	USERDATA
NOTIFY	RETMSGOTOKEN	#TARGETS
	SENDTO	

- The following keyword is supported by version 3 and subsequent versions of the IXCMSGO macro.

MSGACCESS

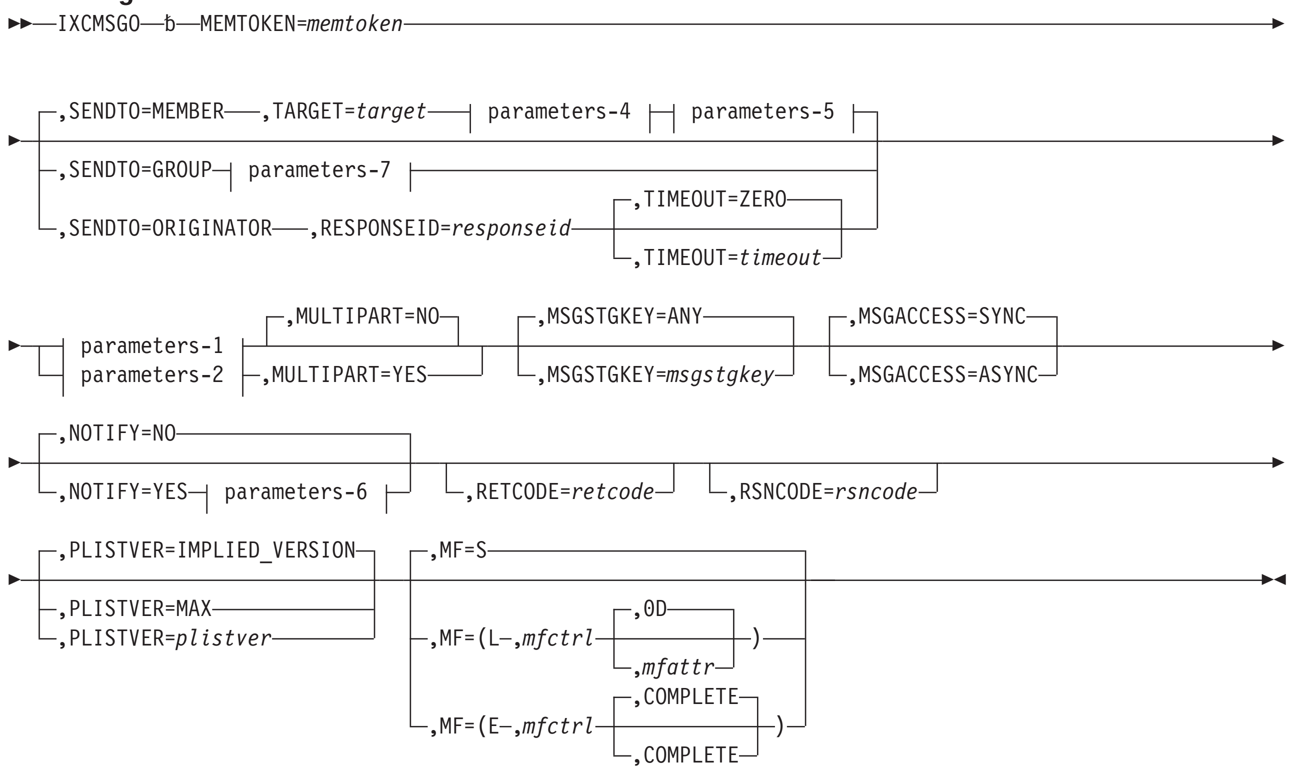
Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

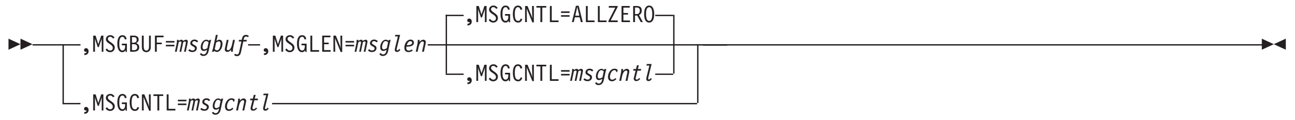
The syntax of the IXCMSGO macro is as follows:

IXCMSGO Macro

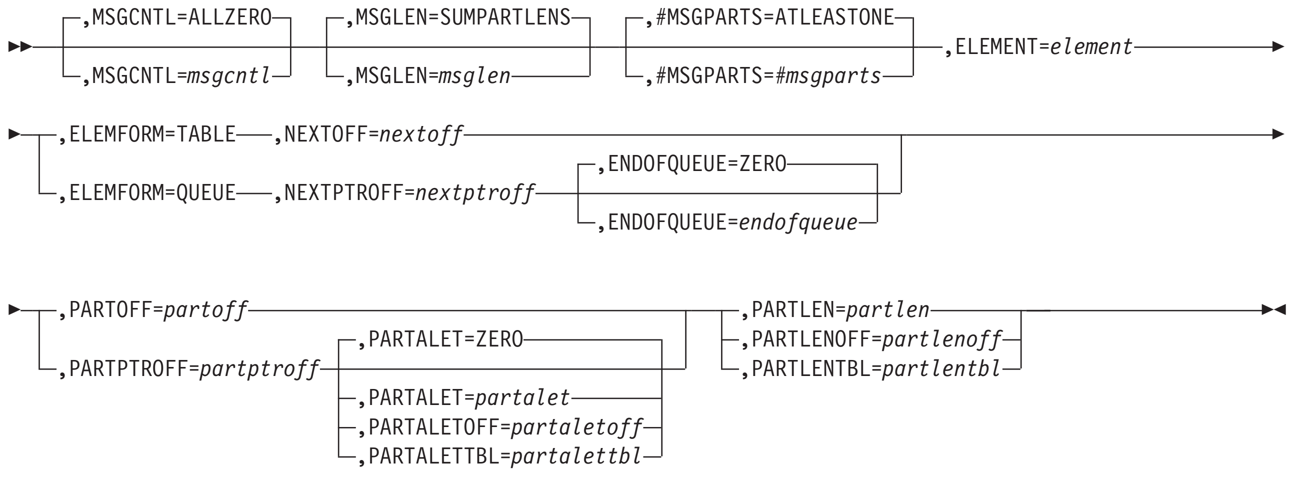
main diagram



parameters-1



parameters-2



parameters-3

```
                                             ┌─,NEXTTAARGETOFF=8──────────────┐
►►──,TARGETS=targets──,#TARGETS=#targets─────┤                                ├──────────────────────────►◄
                                             └─,NEXTTARGETOFF=nexttargetoff───┘
```

parameters-4

```
     ┌─,DELIVERMSG=UNORDERED─────────────────────────┐
►►───┤                                               ├────────────────────────────────────────────────►◄
     │                         ┌─,STREAMID=1───────┐ │
     └─,DELIVERMSG=ORDERED─────┤                   ├─┘
                               └─,STREAMID=streamid┘
```

parameters-5

```
                           ┌─,TIMEOUT=ZERO────┐
     ┌─,GETRESPONSE=NO─────┤                  ├─┐
     │                     └─,TIMEOUT=timeout─┘ │
►►───┤                                          ├──────────────────────────────────────────────────────►◄
     └─,GETRESPONSE=YES─,TIMEOUT=timeout────────┘
```

parameters-6

```
     ┌─,USERDATA=ALLZERO──┐  ┌─,NOTIFYIF=COMPLETED─┐
►►───┤                    ├──┤                     ├──┬─────────────────────────────────────────────────────┬──►
     └─,USERDATA=userdata─┘  └─,NOTIFYIF=FAILED────┘  │                  ┌─,NOTIFYEXIT=FROMJOIN───┐         │
                                                      └─,NOTIFYBY=EXIT───┤                        ├─────────┘
                                                                         └─,NOTIFYEXIT=notifyexit─┘

►──┬────────────────────────────┬──────────────────────────────────────────────────────────────────────►◄
   └─,RETMSGOTOKEN=retmsgotoken─┘
```

parameters-7

```
►►───┬─,MEMBERS=TABLE─┤ parameters-3 ├─┬──┤ parameters-4 ├──┤ parameters-5 ├──────────────────►◄
     ├─,MEMBERS=ALL────────────────────┤
     └─,MEMBERS=OTHER──────────────────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,DELIVERMSG=<u>UNORDERED</u>
,DELIVERMSG=ORDERED

Use this input parameter to indicate whether the system is to deliver the message in sequential order.

DELIVERMSG=UNORDERED

The system is to deliver the message as soon as possible. No ordering is imposed on this message with respect to any other message sent by the sending member to the target member.

Given any other message sent by the sending member to the target member, no assumption can be made about the order in which the system will deliver that message and this message.

DELIVERMSG=ORDERED

The system is to impose ordering on this message with respect to other ordered messages sent by the sending member to the target member. The ordering is determined by the order in which the message-out service accepts the messages for delivery. The system presents the ordered messages to the message user routine of the target member in the same order that the messages were accepted for delivery. Delivery of these messages might be delayed to ensure that proper ordering occurs.

Note that the ordering is imposed on messages sent between a particular **pair** of members. The ordered delivery of messages sent between one particular pair of members is independent of message delivery between any other pair of members.

Use the STREAMID keyword to send messages in independently ordered streams between a particular pair of members. The streams of any one pair of members are independent of the streams of any other pair of members. For a given pair of members, messages within a particular stream are delivered independently of any other stream.

For ordered message delivery to occur, the systems on which the sending member and the target member reside must both be running a release that supports ordered message delivery. If the message is sent to a target member that resides on a release that does not support ordered message delivery, unordered delivery occurs. Use the XCF Query Service (IXCQUERY) to determine whether the ordered delivery protocol is supported for a particular target member.

,ELEMENT=*element*

Use this input parameter to specify the first element of the table or queue of message data elements. Message data elements contain either buffers or pointers to buffers that contain parts of the message to be sent.

Specifying the PARTOFF parameter indicates that each element contains a buffer. Specifying the PARTPTROFF parameter indicates that each element contains a pointer to a buffer.

Elements must all reside in the same address space or data space — the caller's primary address space, an address space or data space that is accessible through a public entry on the caller's dispatchable unit access list (DU-AL), or a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the address of the first element.

,ELEMFORM=TABLE
,ELEMFORM=QUEUE

Use ELEMFORM=TABLE to indicate that the message data elements are organized as a table.

Use ELEMFORM=QUEUE to indicate that the message data elements are organized as a queue.

,ENDOFQUEUE=ZERO
,ENDOFQUEUE=*endofqueue*

Use this input parameter to specify the address that marks the end of the queue. When the pointer to the next element contains this address, queue processing ends.

Note: The queue must have at least one element.

If you omit ENDOFQUEUE, or specify ENDOFQUEUE=ZERO, the default value for the end-of-queue address is zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the end-of-queue address.

,GETRESPONSE=NO
,GETRESPONSE=YES

Use this input parameter to specify whether XCF is to manage the collection of responses to this message.

GETRESPONSE=NO

XCF is not to manage the collection of responses for this message. It is the responsibility of the user to handle any management of responses.

GETRESPONSE=YES

XCF is to manage the collection of responses for this message. When all responses are collected, XCF presents them to the message sender. Note that XCF recognizes situations for which it is not reasonable to expect a response, such as when a target member's system fails.

Response collection is a cooperative effort between XCF and the target members. When XCF presents the message to a target member's message user routine, the parameter list contains the following:

- A flag (MEPLNEEDSRESPONSE in IXCYMEPL) is set to indicate that XCF is managing response collection.
- A response identifier (MEPLRESPONSEID in IXCYMEPL) is provided to allow XCF to identify the message to which the response is to be associated.

After composing the response, the target member replies by invoking the message-out service to send the response to the ORIGINATOR of the message identified by the RESPONSEID. XCF recognizes the message as a response by the particular form of the message-out invocation used to make the reply. When collection of the response(s) has completed, XCF presents the collected response message(s) to the member that originated the GETRESPONSE message-out request through the member's message notify user routine.

For XCF to collect a response, the system on which a target member resides must be running a release that supports XCF-managed response collection. Also, the target member must be capable of participating in this protocol by invoking the message-out service to send its reply in a way that allows XCF to recognize it as a response. XCF sends the message to a target member even if it appears that the member is unable to participate in the XCF response collection. However, XCF does not expect a response from such a member. Use the XCF Query Service (IXCQUERY) to determine whether the XCF-managed response collection protocol is supported for a particular target member.

,MEMBERS=TABLE
| **,MEMBERS=ALL**
| **,MEMBERS=OTHER**
| Use this input parameter to indicate how the collection of target members is to be determined.

| **MEMBERS=TABLE**
| The sender is providing a table that contains a member token for each intended target member.

| The table is an array of entries. Each entry has the same fixed size, and can contain data other
| than a target member token. The storage location identified by the TARGETS keyword contains
| the first 64-bit target member token. Subsequent target member tokens are iteratively located by
| adding the value NEXTTARGETOFF to the address of each member token in turn. The keyword
| #TARGETS indicates the number of entries in the table.

| The table can contain "holes", that is, entries that contain a target member token of X'0'. XCF will
| skip these entries, but will preserve the entries so that the TARGETS table has a one-to-one
| correspondence with any other table that XCF constructs for this request. For example, the "i'th"
| entry of the TARGETS table would correspond to the "i'th" MQATARGRESPENTRY (or
| MQATARGONLYENTRY) returned by the XCF Message Control Query Message service
| (IXCMSGC), or would correspond to the "i'th" MNPLTARGRESPENTRY (or
| MNPLTARGONLYENTRY) presented to a message notify user routine.

| The IXCMSGO invoker is responsible for maintaining the integrity of the TARGETS table until the
| Message-out service returns. If a table changes while being processed by the Message-out
| service, the message might be sent to a different set of targets than expected. Also, the content of
| the entries in tables constructed by XCF might no longer correspond to the content of the entries
| in the TARGET table.

MEMBERS=ALL
Send a copy of the message to every active member in the sender's XCF group (includes the sender).

XCF determines the collection of members that are currently active in the sender's XCF group throughout the sysplex. This collection is equivalent to the list of members that would be returned by invoking the XCF Query service IXCQUERY with REQTYPE=IMMEDIATE.

MEMBERS=OTHER
Send a copy of the message to every active member in the sender's XCF group except for the sender.

XCF determines the collection of members that are currently active in the sender's XCF group throughout the sysplex. This collection is equivalent to the list of members that would be returned by invoking the XCF Query service IXCQUERY with REQTYPE=IMMEDIATE, with the sender excluded.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

MEMTOKEN=*memtoken*
Use this input parameter to specify the member token you received when you issued the IXCJOIN macro to join your XCF group. This token identifies the member sending the message.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-bit field that contains the member token of the sending member.

,MSGACCESS=SYNC
,MSGACCESS=ASYNC

Use this input parameter to indicate how XCF can access the storage containing the text of the message.

MSGACCESS=SYNC

XCF accesses the storage containing the text of the message synchronously with the Message-Out request. After the Message-Out service returns, the sender can dispose of the storage containing the message. For multipart messages, the sender can dispose of the storage containing the elements or tables that define the parts of the message as well.

MSGACCESS=ASYNC

XCF is allowed to access the storage containing the text of the message after the Message-Out request returns to the sender. Note that specifying MSGACCESS=ASYNC does not necessarily imply that XCF will access the storage asynchronously. When possible, XCF will attempt to use synchronous access.

If IXCMSGO returns with return code X'4' and reason code X'410', the sender must preserve the storage containing the message text until the message is completed. For multipart messages, the elements and tables or queues must be preserved as well. For any other return and reason code, the sender can dispose of the storage as if MSGACCESS=SYNC were specified.

If IXCMSGO returns with return code X'0', implying that XCF has accepted the message for delivery, **and** you have specified that the message notify user routine is to receive control when the message is complete, you do not need to preserve the storage containing the message text or the elements and tables or queues. However, the application may have to coordinate responsibility for freeing the message storage area between the sender and the notify exit. The notify exit can determine whether the message storage area had to be preserved by checking the MNPLMSGOASYNCMSGACCESS flag.

If the sending member becomes inactive before an ASYNC message is completed, the Message-Out service might terminate processing for the message. For example, the message might not be delivered to the target member(s) once the sender becomes inactive. In contrast, with MSGACCESS=SYNC, delivery of messages that were accepted by the Message-Out service continues even after the sender becomes inactive.

MSGACCESS=ASYNC must be specified to send message longer than 62,464 (61K) bytes. A nonzero TIMEOUT value also is required when MSGACCESS=ASYNC is specified.

See “Restrictions” on page 137 for additional information about using MSGACCESS=ASYNC.

,MSGBUF=*msgbuf*

Use this input parameter with MULTIPART=NO to specify the buffer containing the message to be sent. The storage key of the buffer specified by MSGBUF must match the storage key specified with the MSGSTGKEY parameter (if you code it).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the storage area containing the message.

,MSGCNTL=ALLZERO
,MSGCNTL=*msgcntl*

Use this input parameter to specify a storage area containing user-specified control information to help the receiving member process the message. Possibilities include:
- The type of message being sent
- The format of the message
- A sequence number to help detect missing signals when using ordered delivery.

The message control information is passed to the message user routine of the receiving member in the message exit parameter list (MEPL), which is mapped by the IXCYMEPL macro.

If you omit the MSGCNTL parameter, or if you specify MSGCNTL=ALLZERO, the message control information presented to the receiving member consists of zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 32-byte storage area containing the message control information.

,MSGLEN=SUMPARTLENS
,MSGLEN=*msglen*
Use this input parameter to specify the length of the message in bytes.

- For MSGACCESS=SYNC, the value must be in the decimal range of 0 to 62464 (61K) bytes.
- For MSGACCESS=ASYNC, the value can be in the decimal range of 0 to 134,217,728 (128M) bytes. If MSGLEN exceeds 61K, both the sending member and the target member must have specified GT61KMSG=YES when the IXCJOIN macro was invoked to join the group. Use the IXCQUERY service to determine whether a particular target member supports the large message delivery protocol.

The amount of buffer storage you provide must be equal to or greater than the value of MSGLEN.

If you specify MULTIPART=YES, then the sum of the lengths of the parts is the default value.

Note: SUMPARTLENS may be specified only when MULTIPART=YES is specified.

If you omit MSGLEN, IXCMSGO determines the message length for you, but performance is reduced.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the length of the message.

,MSGSTGKEY=ANY
,MSGSTGKEY=*msgstgkey*
Use this input parameter to specify the storage key to be used by the system when fetching the message from each buffer. XCF can fetch the message data successfully from a buffer only if the storage key of the buffer matches the storage key specified by MSGSTGKEY or the storage is not fetch protected.

If you omit the MSGSTGKEY parameter, or if you specify MSGSTGKEY=ANY, XCF does not verify that the buffers have any particular storage key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field formatted as **kkkkxxxx**, where the high-order four bits contain the storage key. The low-order halfword is ignored.

,MULTIPART=NO
,MULTIPART=YES
Use MULTIPART=NO to indicate that the message is in a single buffer.

Use MULTIPART=YES to indicate that the message is in one or more buffers.

,NEXTOFF=*nextoff*
Use this input parameter to specify the number of bytes to be added to the address of the current element to obtain the address of the next element. This value equals the size in bytes of an individual message data element. It is used when the message data elements are in table form.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,NEXTPTROFF=*nextptroff*
Use this input parameter to specify the number of bytes to be added to the address of the current

element, to locate within the current element the fullword pointer to the next element. This value is used when the message data elements are in queue form.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,NEXTTARGETOFF=8
,NEXTTARGETOFF=*nexttargetoff*

Use this input parameter to specify, in bytes, the relative location of the next member token in the TARGETS table. To locate the next member token, add the NEXTTARGETOFF value to the location of the current member token. Usually this value is the length of an individual entry in the TARGETS table.

The default value, if NEXTTARGETOFF is not specified, is 8 bytes, meaning that each table entry consists of a member token only.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field containing the number of bytes in an individual table entry.

,NOTIFY=NO
,NOTIFY=YES

Use this input parameter to indicate whether the member is to be notified of message completion.

A message is considered complete in the following circumstances:

- A message-out request times-out or is forced to completion by IXCMSGC with TYPE=COMPLETION.
- XCF has initiated the send of a message without response
- XCF has initiated the send of a broadcast message to each valid target member
- The response to a message with response has arrived.
- The expected responses to a broadcast message with response have arrived.

NOTIFY=NO
Specify this option when you do not require notification of message completion. This option provides compatibility with previous versions of IXCMSGO.

NOTIFY=YES
Specify this option when you require notification of message completion. XCF will maintain status information about the message until the message is completed. Upon completion of the message, the state of the message and the NOTIFYIF keyword determine how XCF is to proceed. If a member is to be notified of message completion, XCF preserves status information and uses the NOTIFYEXIT to inform the member.

,NOTIFYBY=EXIT

Use this input parameter to indicate that the member is to be notified of message completion through a message notify user routine scheduled by XCF.

,NOTIFYEXIT=FROMJOIN
,NOTIFYEXIT=*notifyexit*

Use this input parameter to identify the message notify user routine to receive control when the message is complete. If omitted, the system uses the message notify user routine specified on IXCJOIN when the member joined the XCF group. Note that the systems rejects the IXCMSGO request if the NOTIFYEXIT defaults to FROMJOIN and no message notify user routine was specified on IXCJOIN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 31-bit message notify user routine.

,NOTIFYIF=COMPLETED
,NOTIFYIF=FAILED

Use this input parameter to identify the conditions under which the system is to provide notification of message completion. See the NOTIFY keyword for the definition of message completion.

NOTIFYIF=COMPLETED
The system is to provide notification when the message is completed. Notification occurs whether the message succeeded or failed.

NOTIFYIF=FAILED
The system is to provide notification only if the message failed.

- A message without response is considered to have failed if the message was not sent to one of the possible target members.
- A message with response is considered to have failed if XCF does not receive a response from every possible target member. (For example, if a message with response is targetted to a member that is no longer active — and therefore, no response is possible — the message is considered to have failed.

  Note that skipped targets are ignored when determining whether the message failed. A skipped target is one whose member token in the TARGETS table is X'0'.

,PARTALET=ZERO
,PARTALET=*partalet*
Use this input parameter to specify a single ALET to be used to qualify every buffer address. The ALET must be zero, a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

If you omit PARTALET, PARTALETOFF, and PARTALETTBL, the default is PARTALET with an ALET of zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the ALET.

,PARTALETOFF=*partaletoff*
Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword field that contains the ALET to be used to qualify the buffer address associated with that element. The ALET must be zero, a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTALETTBL=*partalettbl*
Use this input parameter to specify a table in which each entry is a fullword containing the ALET to qualify a buffer address in the table or queue of message data elements. For instance, the 3rd entry in PARTALETTBL must contain the ALET to qualify the buffer address in the 3rd message data element. *partalettbl* must begin on a fullword boundary. Each ALET must be zero, a public entry on your dispatchable unit access list (DU-AL), or an entry for a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of *partalettbl*.

,PARTLEN=*partlen*
Use this input parameter to specify the length in bytes of each buffer (element) when all the buffers are the same length. The length of the entire message is equal to PARTLEN multiplied by the number of elements. All buffers must be at least as long as PARTLEN.

- When MSGACCESS=SYNC, PARTLEN must be in the decimal range of 0 to 62464 (61K) bytes. The length of the entire message must be in the decimal range of 0 to 62464 bytes.
- When MSGACCESS=ASYNC, PARTLEN can be in the decimal range of 0 to 134,217,728 (128M) bytes. The length of the entire message must be in the decimal range of 0 to 128M bytes. If the length of the entire message exceeds 61K bytes, both the sending member and the target member must have specified GT61KMSG=YES when they invoked the IXCJOIN macro to join the group. Use the IXCQUERY service to determine whether a particular target member supports the large message delivery protocol.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the length in bytes of each buffer.

,PARTLENOFF=*partlenoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword field that contains the length in bytes of the buffer associated with that element. The length of each buffer can range from 0 to 62464 (61K) bytes in decimal when MSGACCESS=SYNC is specified and from 0 to 134,217,728 (128K) bytes in decimal when MSGACCESS=ASYNC is specified.

If you specify the length of a buffer as zero, XCF does not use that buffer.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTLENTBL=*partlentbl*

Use this input parameter to specify a table in which each entry is a fullword containing the length of a corresponding buffer. For instance, the 3rd entry in PARTLENTBL contains the length of the buffer whose address is associated with the 3rd message data element in the table or queue.

- When MSGACCESS=SYNC, the length of an individual message part must be in the decimal range of 0 to 62464 (61K) bytes. The length of the entire message must be in the decimal range of 0 to 62464 bytes.
- When MSGACCESS=ASYNC, the length of an individual message part must be in the decimal range of 0 to 134,217,728 (128M) bytes. The length of the entire message must be in the decimal range of 0 to 128M bytes. If the length of the entire message exceeds 61K bytes, both the sending member and the target member must have specified GT61KMSG=YES when they invoked the IXCJOIN macro to join the group. Use the IXCQUERY service to determine whether a particular target member supports the large message delivery protocol.

If you specify the length of a buffer as zero, XCF does not use that buffer.

The table specified by PARTLENTBL must begin on a fullword boundary.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of *partlentbl*.

,PARTOFF=*partoff*

Use this input parameter to specify the number of bytes to be added to the address of the current element to obtain the address within the element of the start of the buffer associated with it.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PARTPTROFF=*partptroff*

Use this input parameter to specify the number of bytes to be added to the address of the current element, to locate within the element the fullword pointer to the buffer associated with it.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of bytes.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXCMSGO Version Support" on page 138 for a description of the options available with PLISTVER.

,RESPONSEID=*responseid*

Use this input parameter to provide the XCF identifier for the originating message to which this request is making a response.

Obtain the RESPONSEID from one of the following:

- The MEPLRESPONSEID field in the IXCYMEPL that was presented to a message user routine.

- The MQAMIDRESPONSEID field in the IXCYMQAA that was returned by the XCF Message Control service (IXCMSGC REQUEST=QUERYMSG).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 24-byte character field containing the XCF response identifier.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RETMSGOTOKEN=*retmsgotoken*
Use this output parameter to specify a storage area to contain a token that you can use to identify this message to the XCF Message Control service (IXCMSGC).

The storage area must be in the caller's primary address space, in an address space or data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL), or in a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte character output field to contain the token.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,SENDTO=MEMBER
,SENDTO=GROUP
,SENDTO=ORIGINATOR
Use this input parameter to specify the member(s) to which the message should be sent.

SENDTO=MEMBER
Send the message to the (one) member indicated by TARGET. This type of send is the default.

SENDTO=GROUP
Send the message to a collection of members. This type of send is referred to as a "broadcast".

Note that a send to GROUP where the collection of members consists of only one member does not necessarily have the same behavior as a send to MEMBER. For example, suppose that in both cases the one target member specified was not a valid target. The system would reject the send to MEMBER request with return code X'08', reason code X'08', indicating that the target was not valid. The system would reject the send to GROUP request with return code X'04', reason code X'404', indicating that the broadcast completed but the send to at least one target was rejected. The send reason code reported for the individual target in the SENDTO=GROUP collection would indicate that the target was not valid.

SENDTO=ORIGINATOR
Send a response message. This is a response to a previously received message that indicated that the sender had requested that XCF manage the gathering of response(s) to that message.

,STREAMID=1
,STREAMID=*streamid*
Use this input parameter to identify the stream of ordered messages to which this message belongs. STREAMID must be a decimal value between 1 and 15, inclusive. The default value is a STREAMID of 1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field containing the stream identification.

,TARGET=*target*

Use this input parameter to specify the member token of the member to receive the message. See *z/OS MVS Programming: Sysplex Services Guide* for information about how to obtain the member token of the receiving member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-bit field that contains the member token.

,TARGETS=*targets*

Use this input parameter to specify the storage area containing the table of member tokens for each intended target member. The storage location named by TARGETS contains the first member token to be processed.

The storage area identified by TARGETS must be in the caller's primary address space, in an address space or data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL), or in a common area data space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the storage area containing the table of member tokens for each intended target member.

,TIMEOUT=ZERO
,TIMEOUT=*timeout*

Use this input parameter to specify the number of seconds the user is willing to allow for the message to complete. (See the NOTIFY keyword for a definition of message completion.) The timeout value specified must be between 1 and 32,767 (X'7FFF').

If the message has not completed before the expiration of the timeout value, the message is declared to be complete. The system then processes the message completion as specified by the NOTIFY keyword.

A nonzero timeout value is required when MSGACCESS=ASYNC is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword field containing the timeout value.

,USERDATA=ALLZERO
,USERDATA=*userdata*

Use this input parameter to specify eight characters of user data to be associated with the message. The system presents this data in the message notification parameter list (IXCYMNPL) when the message is presented to the message notify user routine. The user data is also available through the IXCMSGC REQUEST=QUERYMSG service with DATATYPE=MSGOUT.

The user data is not presented to the targets of the message.

The default of ALLZERO consists of hexadecimal zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the eight-character field containing the user data.

,#MSGPARTS=ATLEASTONE
,#MSGPARTS=*msgparts*

Use this input parameter with MULTIPART=YES to specify the number of buffers (1 or more) containing the message data to be sent.

If you specify the default value (#MSGPARTS=ATLEASTONE), one message part is the minimum, or more as needed to send MSGLEN bytes.

If you omit #MSGPARTS, the result depends on the following:

- If you specify MSGLEN, the buffer storage you provide must be equal to or greater than the value of MSGLEN.
- If you omit MSGLEN and specify a table of message data elements (ELEMFORM=TABLE), XCF processes only the first message data element.

- If you omit MSGLEN and specify a queue of message data elements (ELEMFORM=QUEUE), XCF processes message data elements until it reaches the end of the queue. See the ENDOFQUEUE parameter description to find out how XCF detects the end of the queue.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword containing the number of buffers.

,#TARGETS=*#targets*

Use this input parameter to specify the number of entries in the TARGETS table. The value specified must be between 1 and the maximum number of members per XCF group supported by the sysplex, inclusive.

The system programmer determines the maximum number of members per XCF group supported by the sysplex when using the XCF format utility (IXCL1DSU) to create the sysplex couple data set for the sysplex. In effect, the message can be broadcast to every member that can successfully invoke IXCJOIN to join the XCF group.

Note that #TARGETS indicates the number of potential targets. A TARGETS table entry that contain a member token of X'0' is considered a potential target and should be included in the #TARGETS count.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field containing the number of entries in the TARGET table.

ABEND Codes

Abend codes an issuer of IXCMSGO might receive are listed below. For detailed abend code information, see *z/OS MVS System Codes*.

X'073'
X'C78'

Return and Reason Codes

When the IXCMSGO macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 12. Return and Reason Codes for the IXCMSGO Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
00	None.	**Meaning:** IXCMSGO completed successfully; XCF accepts the message for delivery. **Action:** None.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
04	401	**Equate Symbol:** IXCMSGORSNSENDPENDING **Meaning:** Send message pending. The system could not initiate the send to the requested target member immediately and the send is pending. XCF continues trying to send the message until the specified TIMEOUT value expires. **Action:** Discontinue sending the message until the condition clears. The system will drive the message notify user routine when the send completes if notification was requested.
04	402	**Equate Symbol:** IXCMSGORSNBCPENDINGNOREJECTS **Meaning:** Broadcast pending, no rejections. The system could not initiate the send to one or more target members immediately and the send is pending. No send was rejected. XCF continues trying to send the pending message(s) until the specified TIMEOUT value expires. **Action:** Discontinue sending the message until the condition clears. The system will drive the message notify user routine when the send completes.
04	403	**Equate Symbol:** IXCMSGORSNBCPENDINGWITHREJECTS **Meaning:** Broadcast pending, some send(s) rejected. The system could not initiate the send to one or more target members immediately and the send is pending. The send for at least one target member was rejected. XCF continues trying to send the pending message(s) until the specified TIMEOUT value expires. **Action:** Discontinue sending the message until the condition clears. The system will drive the message notify user routine when the send completes.
04	404	**Equate Symbol:** IXCMSGORSNBCCOMPLETEWITHREJECTS **Meaning:** Broadcast completed, some send(s) rejected. No sends to target members are pending. The send for at least one target member was rejected. **Action:** None.
04	0405xxxx	**Equate Symbol:** IXCMSGORSNRETMSGOTOKENNOACCESS **Meaning:** RETMSGOTOKEN is not accessible. IXCMSGO was unable to store a message token in the area indicated by RETMSGOTOKEN. The low order halfword (xxxx) of the reason code has a value of zero if the return code was X'00' or indicates the reason code associated with return code X'04' that the system would have returned if the storage area indicated by RETMSGOTOKEN had been accessible. Note that even though the system was unable to store the RETMSGOTOKEN token, if the return code is X'00', the system has initiated the send of the message, or if the return code is X'04', the system has accepted the message for delivery. **Action:** Verify that the storage area indicated by RETMSGOTOKEN is either in the caller’s primary address space or in an address or data space that is addressable through a public entry on the caller’s dispatchable unit address list (DU-AL) or is in a common area data space.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
04	410	**Equate Symbol:** IXCMSGORSNASYNCSENDPENDING **Meaning:** A send message is pending. The send to a requested target member could not be initiated immediately and is pending. XCF will continue trying to send the message until the specified TIMEOUT value expires. This reason code is applicable only when MSGACCESS=ASYNC is specified. **Action:** Preserve the message text until the send of the message is complete. For a multipart message, the queue or table elements that describe how to locate the message must also be preserved. If NOTIFY=YES is specified, the system will The system will drive the message notify user routine when the message is completed.
08	04	**Equate Symbol:** IXCMSGORSNSENDERNOTVALID **Meaning:** Program error. One of the following happened: • The sending member token does not identify an active member associated with the primary address space that was current when the IXCMSGO service was invoked. • The primary address space is not the master scheduler's address space. • A member has terminated or placed itself in a not-defined state. **Action:** Take one or more of the following actions: • Ensure that the correct sending member token was specified. • Ensure that the parameter list was not inadvertently overlaid and that it was correctly specified. • Ensure that the IXCMSGO service is being called either from the address space that was current when the sending member issued IXCJOIN to join its group or from the master scheduler address space. Any further action depends on your application.
08	08	**Equate Symbol:** IXCMSGORSNTARGETNOTVALID **Meaning:** Program error. The token of the target member either: • Represents a member in a different XCF group. • Represents a member that is not active. • Is not a valid XCF member token. **Action:** Take one or more of the following actions: • Ensure that the target and source member tokens represent members of the same XCF group. Messages cannot be sent between members of different groups. • If the target member is not active, any action taken depends on your application. • Ensure that the correct target member token was specified. • Ensure that the parameter list was not inadvertently overlaid and that it was correctly specified.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	0C	**Equate Symbol:** IXCMSGORSNMSGLENNOTVALID **Meaning:** Program error. For MSGACCESS=SYNC, the total message length is not in the decimal range of 0 to 62464. For MSGACCESS=ASYNC, the total message length is not in the decimal range of 0 to 134,217,728 or the sending member did not specify YES for the GT61KMSG keyword when it invoked IXCJOIN to join the group. **Action:** Take one or more of the following actions: • Ensure that the correct message length was specified. • The maximum message size is 62464 bytes. If your message is larger than 62464 bytes, you will need to break it up into parts no larger than 62464, and send each of them separately. • Ensure that the parameter list was not inadvertently overlaid and that it was correctly specified.
08	10	**Equate Symbol:** IXCMSGORSNMSGBUFBADSTG **Meaning:** Program error. XCF could not access the message buffer. **Action:** Take one or more of the following actions: • Ensure that the correct message buffer storage area was used. • If your program is running in AR mode, ensure that: – The ALET for the message buffer is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • Ensure that the message buffer storage area was not inadvertently freed by your program.
08	14	**Equate Symbol:** IXCMSGORSNMSGCNTLBADALET **Meaning:** Program error. The address of the message control information (MSGCNTL) is qualified by an ALET that is not valid. The ALET must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Take one or more of the following actions: • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • Ensure that the ALET for the message control information is correct.
08	18	**Equate Symbol:** IXCMSGORSNMSGCNTLBADSTG **Meaning:** Program error. The system cannot access the area that contains the message control information (MSGCNTL). **Action:** Take one or more of the following actions: • Ensure that the address of the message control information is correct. • If your program is running in AR mode, ensure that: – The ALET for the message control information is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXCMSGO macro.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	1C	**Equate Symbol:** IXCMSGORSNTARGETNOMSGEXIT **Meaning:** Program error. The member specified on the TARGET parameter does not have a message user routine. **Action:** Take one or more of the following actions: • Ensure that the correct TARGET member token was specified. • Ensure that the TARGET member specified a message user routine (MSGEXIT keyword on IXCJOIN) when joining the XCF group.
08	40	**Equate Symbol:** IXCMSGORSNPLISTRSVDNOTVALID **Meaning:** Program error. A reserved field in the control parameter list was not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of z/OS on which your program is running.
08	100	**Equate Symbol:** IXCMSGORSNPLISTBADALET **Meaning:** Program error. The ALET that qualifies the address of the control parameter list was not valid. The ALET must be zero. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • The ALET for the control parameter list storage is correct.
08	104	**Equate Symbol:** IXCMSGORSNPLISTVERSIONNOTVALID **Meaning:** Program error. The version number in the control parameter list was not valid. **Action:** Check for errors such as the following: • Your program overlaid the control parameter list storage. • Your program was assembled with the wrong macro library for the release of z/OS on which your program is running.
08	10C	**Equate Symbol:** IXCMSGORSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Ensure that: • The correct parameter list storage area was specified. • The parameter list storage area was not inadvertently freed by your program.
08	11C	**Equate Symbol:** IXCMSGORSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it does not issue IXCMSGO while it is disabled.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	12C	**Equate Symbol:** IXCMSGORSNLOCKHELD **Meaning:** Program error. The caller of IXCMSGO holds a lock. **Action:** Correct your program so that it does not issue IXCMSGO while it is holding a lock.
08	208	**Equate Symbol:** IXCMSGORSNMSGBUFBADALET **Meaning:** Program error. The ALET that qualifies the address of the buffer specified by MSGBUF must be zero, a valid public entry on your DU-AL, or a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for the buffer.
08	20C	**Equate Symbol:** IXCMSGORSNMSGBUFKEYMISMATCH **Meaning:** Program error. The service could not fetch the message data from the buffer area specified by MSGBUF using the storage key specified by MSGSTGKEY. **Action:** Check for errors such as the following: • The buffer area was allocated in storage whose storage key does not match the key specified by the MSGSTGKEY parameter. • The storage key specified by MSGSTGKEY has been overlaid • The storage key specified by MSGSTGKEY is not valid. • The address of the buffer area is incorrect.
08	210	**Equate Symbol:** IXCMSGORSNPARTPTROFFBADSTG **Meaning:** Program error. An element is not accessible. The address located within an element at the offset specified by PARTPTROFF was not valid. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the invalid address. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the invalid address. **Action:** Check for errors such as the following: • You specified an incorrect element address. • Your program overlaid the element address. • The buffer had been previously freed. • You specified an incorrect number of message parts. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value. • If your program is running in AR mode, check for the following: – You specified the wrong ALET for the element. – You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	212	**Equate Symbol:** IXCMSGORSNELEMENTBADALET **Meaning:** Program error. The ALET that qualifies the address of an ELEMENT must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for the buffer.
08	213	**Equate Symbol:** IXCMSGORSNNEXTPTROFFBADSTG **Meaning:** Program error. A pointer to the next queue element, at the offset specified by NEXTPTROFF, was not valid. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the incorrect address. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the incorrect address. **Action:** Check for errors such as the following: • You specified an incorrect element address. • Your program overlaid the element address. • You specified an incorrect number of message parts. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.
08	214	**Equate Symbol:** IXCMSGORSN#MSGPARTSZERO **Meaning:** Program error. The value specified by the #MSGPARTS parameter was zero but must be greater than zero. **Action:** Specify a value greater than zero for #MSGPARTS.
08	215	**Equate Symbol:** IXCMSGORSNTOOMANYZEROLENPARTS **Meaning:** Program error. The message data element table or queue is assumed to be damaged. You omitted the #MSGPARTS parameter, and XCF processed more than 65,536 consecutive elements with buffers of length zero. **Action:** Check for errors such as the following: • You specified PARTLEN with a value of zero. • You specified PARTLENTBL but the address of the table was not correct. • You specified PARTLENOFF with an incorrect value. • Ensure that the elements have not been overlaid. • You specified an incorrect end-of-queue value. Specify the #MSGPARTS parameter when invoking IXCMSGO.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	218	**Equate Symbol:** IXCMSGORSNPARTPTROFF@BADSTG **Meaning:** Program error. A buffer whose address was at the offset specified by PARTPTROFF could not be accessed. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the incorrect address. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the incorrect address. **Action:** Check for following types of errors: • The buffer address was incorrect. • The buffer was previously freed. • You specified PARTPTROFF with an incorrect value. • If your program is running in AR mode, check for the following types or errors: – You specified the wrong ALET for the buffer. – You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro.
08	219	**Equate Symbol:** IXCMSGORSNPARTOFFBADSTG **Meaning:** Program error. A buffer located at the offset specified by PARTOFF could not be accessed. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the buffer that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer that could not be accessed. **Action:** Check for following types of errors: • The storage containing the element has been freed. • The PARTOFF value is incorrect. • The storage containing the element has been overlaid. • You specified an incorrect number of message parts. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	21C	**Equate Symbol:** IXCMSGORSNPARTPTROFF@KEYMISMATCH **Meaning:** Program error. XCF could not fetch the message data from a buffer whose address was at the offset specified by PARTPTROFF using the storage key specified by MSGSTGKEY. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the address of the buffer from which the message data could not be fetched. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the address of the buffer from which the message data could not be fetched. **Action:** Check for errors such as the following: • The buffer was allocated in storage whose storage key did not match the key specified by the MSGSTGKEY parameter. • You specified PARTPTROFF with an incorrect value. • The address of the buffer is not correct.
08	21D	**Equate Symbol:** IXCMSGORSNPARTOFFKEYMISMATCH **Meaning:** Program error. XCF could not fetch the message data from a buffer area located at the offset specified by PARTOFF using the storage key specified by MSGSTGKEY. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the buffer from which the message data could not be fetched. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer from which the message data could not be fetched. **Action:** Check for errors such as the following: • The buffer area was allocated in storage whose storage key did not match the key specified by the MSGSTGKEY parameter.
08	220	**Equate Symbol:** IXCMSGORSNPARTLENTBLBADSTG **Meaning:** Program error. The table specified by PARTLENTBL could not be accessed. **Action:** Check for errors such as the following: • PARTLENTBL was inadvertently freed. • The element was overlaid. • The address of the table specified by PARTLENTBL is not correct.
08	221	**Equate Symbol:** IXCMSGORSNPARTLENTBLNOTWORDBDY **Meaning:** Program error. The table specified by PARTLENTBL does not begin on a fullword boundary. **Action:** Ensure that the table specified by PARTLENTBL begins on a fullword boundary.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	222	**Equate Symbol:** IXCMSGORSNPARTLENTBLBADALET **Meaning:** Program error. The ALET that qualifies the address of PARTLENTBL must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for PARTLENTBL.
08	223	**Equate Symbol:** IXCMSGORSNPARTLENOFFBADSTG **Meaning:** Program error. The buffer length located at the offset specified by PARTLENOFF could not be accessed within an element. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the buffer length that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer length that could not be accessed. **Action:** Check for errors such as the following: • The storage containing the element has been freed. • The PARTLENOFF value is incorrect. • The storage containing the element has been overlaid. • The message data elements were not properly constructed or serialized. • You specified an incorrect number of message parts. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.
08	224	**Equate Symbol:** IXCMSGORSNMSGLENGTSUMPARTLEN **Meaning:** Program error. The message length specified by MSGLEN was not valid. The message length specified by MSGLEN is larger than the sum of the buffer lengths. **Action:** Ensure that the message length specified by MSGLEN is less than or equal to the sum of the buffer lengths.
08	225	**Equate Symbol:** IXCMSGORSNPARTLENBADLEN **Meaning:** Program error. The buffer length specified by PARTLEN was not valid. The value must be less than or equal to 62464 decimal. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the buffer length that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer length that was not valid. **Action:** Ensure that the value specified by PARTLEN is less than or equal to 62464.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	226	**Equate Symbol:** IXCMSGORSNPARTLENTBLBADLEN **Meaning:** Program error. A buffer length contained in the table specified by PARTLENTBL was not valid. Buffer lengths must be less than or equal to 62464 decimal. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the buffer length that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the buffer length that was not valid. **Action:** Ensure that all buffer lengths are less than or equal to 62464.
08	227	**Equate Symbol:** IXCMSGORSNPARTLENOFFBADLEN **Meaning:** Program error. The buffer length located at the offset specified by PARTLENOFF was not valid. Buffer lengths must be less than or equal to 62464 decimal. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the invalid buffer length. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the invalid buffer length. **Action:** Ensure that the buffer length is less than or equal to 62464.
08	230	**Equate Symbol:** IXCMSGORSNPARTALETTBLBADSTG **Meaning:** Program error. The table specified by PARTALETTBL could not be accessed. **Action:** Check for errors such as the following: • PARTALETTBL was inadvertently freed. • PARTALETTBL was overlaid or incorrect.
08	231	**Equate Symbol:** IXCMSGORSNPARTALETTBLNOTWORDBDY **Meaning:** Program error. The table specified by PARTALETTBL does not begin on a fullword boundary. **Action:** Ensure that the table specified by PARTALETTBL begins on a fullword boundary.
08	232	**Equate Symbol:** IXCMSGORSNPARTALETTBLBADALET **Meaning:** Program error. The ALET that qualifies the address of PARTALETTBL must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for PARTALETTBL.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	233	**Equate Symbol:** IXCMSGORSNPARTALETOFFBADSTG **Meaning:** Program error. The ALET located at the offset specified by PARTALETOFF could not be accessed. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the ALET that could not be accessed. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that could not be accessed. **Action:** Check for errors such as the following: • The storage containing the element has been freed. • The PARTALETOFF value is incorrect. • The storage containing the element has been overlaid. • The message data elements were not properly constructed or serialized.
08	234	**Equate Symbol:** IXCMSGORSNPARTALET@BADALET **Meaning:** Program error. The ALET specified by PARTALET was not valid. It must be zero, a public entry on your DU-AL, or an entry for a common area data space. **Action:** Correct the ALET.
08	235	**Equate Symbol:** IXCMSGORSNPARTALETTBL@BADALET **Meaning:** Program error. An ALET specified in a PARTALETTBL entry was not valid. Each ALET must be zero, a public entry on your DU-AL, or an entry for a common area data space. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the ALET that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that was not valid. **Action:** Correct the ALET.
08	236	**Equate Symbol:** IXCMSGORSNPARTALETOFF@BADALET **Meaning:** Program error. An ALET specified at the offset designated by PARTALETOFF was not valid. It must be zero, a public entry on your DU-AL, or an entry for a common area data space. In the parameter list generated by IXCMSGO: • The field, PART#, indicates the index of the element containing the ALET that was not valid. PART# values start with 1. • The field, ELEMENTPTR, contains the address of the element containing the ALET that was not valid. **Action:** Correct the ALET or check for errors such as the following: • The PARTALETOFF value is incorrect. • You specified an incorrect number of message parts. • The message data element is damaged or has not maintained the integrity of its data. • You specified an incorrect end-of-queue value.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	300	**Equate Symbol:** IXCMSGORSNSENDERNONOTIFYEXIT **Meaning:** Program error. NOTIFYEXIT is not defined. The value of NOTIFYEXIT defaulted to the address of the exit routine defined when the member joined the group, but no NOTIFYEXIT was defined when the IXCJOIN service was invoked. **Action:** Code the NOTIFYEXIT explicitly or add the NOTIFYEXIT definition to the IXCJOIN invocation.
08	304	**Equate Symbol:** IXCMSGORSNTARGETSBADALET **Meaning:** Program error. The ALET that qualifies the address of the TARGETs table is neither zero or a valid entry on the caller's Dispatchable Unit Access List (DU-AL) nor a valid entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for TARGETS.
08	308	**Equate Symbol:** IXCMSGORSNRETMSGOTOKENBADALET **Meaning:** Program error. The ALET that qualifies the address of the RETMSGOTOKEN is neither zero or a valid entry on the caller's Dispatchable Unit Access List (DU-AL) nor a valid entry for a common area data space. **Action:** Check for errors such as the following: • You did not specify SYSSTATE ASCENV=AR before issuing the IXCMSGO macro. • You specified the wrong ALET for RETMSGOTOKEN.
08	30C	**Equate Symbol:** IXCMSGORSNBADRESPONSEID **Meaning:** Program error. The RESPONSEID is not valid. The RESPONSEID token has been corrupted. **Action:** Ensure that the RESPONSEID token that was provided in the message exit parameter list (MEPL) is the token you are using to respond to the message. Also verify that the storage containing the RESPONSEID token has not been inadvertently freed or overlaid.
08	310	**Equate Symbol:** IXCMSGORSNBADSTREAMID **Meaning:** Program error. The value specified for STREAMID is not valid. STREAMID must contain a decimal value in the range of 1 to 15 inclusive. **Action:** Verify that STREAMID contains a value in the range of 1-15 (decimal).

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	314	**Equate Symbol:** IXCMSGORSNTARGETSBADSTG **Meaning:** Program error. The TARGETS table is not accessible. **Action:** Check for errors such as the following: • TARGETS was inadvertently freed. • TARGETS was overlaid or incorrect.
08	320	**Equate Symbol:** IXCMSGORSNBAD#TARGETS **Meaning:** Program error. The value specified for #TARGETS is not valid. The number of targets must be a nonzero value less than or equal to the maximum number of members per group supported by the sysplex. Note that the maximum number of members per XCF group is determined by the system programmer when the XCF format utility (IXCL1DSU) is used to create the sysplex couple data set that is being used by the sysplex. #TARGETS cannot be larger than the maximum number of members permitted to join the group. **Action:** Correct the #TARGETS value to be within the allowable range. Use the IXCQUERY service to determine the number of members currently in the group.
08	324	**Equate Symbol:** IXCMSGORSNBADTIMEOUT **Meaning:** Program error. The value specified for TIMEOUT is not valid. The timeout value must be in the range of 1 to X'7FFF', inclusive. If MSGACCESS=ASYNC is coded, a nonzero timeout value must be provided. **Action:** Verify that the value specified for TIMEOUT is in the range of 1 to 32,767 (X'7FFF'), inclusive.
08	340	**Equate Symbol:** IXCMSGORSNTARGETMAXMSGLEN61K **Meaning:** Program error. The message length is not valid for the target. Either the target member or the system on which the target member resides does not support messages larger than 62462 bytes (decimal). The message was not sent. **Action:** Use the IXCQUERY service to determine which members reside on systems capable of sending or receiving messages larger than 62462 bytes. For XCF to deliver a message greater than 62462 bytes in length, the target member must reside on a system that supports 128MB message delivery, and the target member must specify GT61KMSG=YES when it invokes the IXCJOIN macro to join its group.
08	344	**Equate Symbol:** IXCMSGORSNSENDERBECAMEINACTIVE **Meaning:** The sending member became inactive during the message-out request. The message-out request is terminated. Some, none, or all of the targets may receive the message. **Action:** None.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	010Cxxxx	**Equate Symbol:** IXCMSGORSNPLISTNOPARTINFOBADSTG **Meaning:** Program error. A parameter specifying information on a request with MULTIPART=YES was not valid. The system was unable to store the erroneous information in PART# and ELEMENTPTR fields in the parameter list generated by IXCMSGO because it could not access the parameter list. The low-order halfword of the reason code (xxxx) contains the reason code for the error that would have been returned if the parameter list had been accessible. (The reason code applies to a X'8' return code.) **Action:** The control parameters must be in the primary address space. Make sure the ALET for the control parameter list is correct.
0C	04	**Equate Symbol:** IXCMSGORSNNOBUFFER **Meaning:** Environmental error. The signalling facility is busy; message buffers are temporarily unavailable. Some of the reasons message buffer space might not be available are: • There was suddenly a large amount of message buffer space usage, which caused all the buffer space to be temporarily exhausted. • There is a lot of competition for message buffer space. It is possible that the installation should have allocated more message buffers for your particular transport class. The installation can use the message buffer limit to control how much of the total message buffer resource your application can use. • Before running or installing your application, you should inform the system programmer of any XCF resources you might require. **Action:** • Try sending the message again after a short period of time. • Invoke IXCMSGO again specifying the TIMEOUT parameter so that XCF will queue the message and resend it automatically when the condition clears. If this condition reoccurs when a nonzero TIMEOUT is specified, XCF will either: – Accept the message with return code X'4', or – Reject the message with return code X'C',reason code X'C' • Inform the system programmer of the space constraints.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	08	**Equate Symbol:** IXCMSGORSNNOPATH **Meaning:** Environmental error. All signalling paths to the target member's system are temporarily unavailable. There could be a problem with the target member's system. **Action:** • Try sending the message again after a short period of time. If all paths are still unavailable, something might be wrong with the target member's system. The member can keep trying periodically to send the message until the target member is no longer active (return code 8, reason code 8) or the member group exit routine receives a system-status-update-missing or system-going notification for the target member. A system-status-update-missing notification indicates that the member's system is stopped for as long as the member's failure detection interval. If a system-status-update-missing notification is received, the member can try again when its group exit routine receives the system status update resumed notification. Once a system-going notification is received, the member's system is terminating. At that point, there is no need to send the message. • Invoke IXCMSGO again specifying the TIMEOUT parameter so that XCF will queue the message and resend it automatically when the condition clears. If this condition reoccurs when a nonzero TIMEOUT is specified, XCF will either: – Accept the message with return code X'4', or – Reject the message with return code X'C', reason code X'i8' See *z/OS MVS Programming: Sysplex Services Guide* for more information.
0C	0C	**Equate Symbol:** IXCMSGORSNNOMSGSPACE **Meaning:** Environmental error. The message space managed by XCF on behalf of the member has no more capacity. Generally this condition suggests that the volume of message traffic has exceeded processing capacity of the member or its group. Depending on how the member is using the signalling service, member message space might become available as the system continues to process work. The member might use the IXCMSGC service with REQUEST=COMPLETION, TYPE=FORCE to attempt to have the message notify user routine process the message and not save it, thus making more member message space storage available. The member might need to take steps to reduce its message traffic volume. **Action:** • Retry the request after allowing some time for the condition to clear, or after taking appropriate actions to release member message space. • Try to free up space by discarding saved messages or pending messages.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	10	**Equate Symbol:** IXCMSGORSNSYSTEMNOSTORAGE **Meaning:** Environmental error. A shortage of system storage has occurred. The IXCMSGO request is rejected. **Action:** Retry the request after allowing some time • Retry the request after allowing some time for the condition to clear. Subsequent requests might also be rejected until the storage shortage is relieved. • The member might use the IXCMSGC service with REQUEST=COMPLETION, TYPE=FORCE to attempt to have the message notify user routine process the message and not save it, thus making more member message space storage available. • Try to free up space by discarding saved messages or pending messages.
0C	14	**Equate Symbol:** IXCMSGORSNNOBUFFERNOTQUEUED **Meaning:** Environmental error. Signalling delivery has been making no progress delivering messages to the target because XCF message buffers used for signalling are unavailable. The IXCMSGO request is rejected and is not queued. The system also rejects any further requests that require queueing until the condition is resolved. **Action:** Retry the request after allowing some time for the condition to clear.
0C	18	**Equate Symbol:** IXCMSGORSNNOPATHNOTQUEUED **Meaning:** Environmental error. XCF signalling paths are unavailable because of a lack of connectivity to the target system. The system rejects the IXCMSGO request and does not queue it. The system also rejects subsequent requests until the condition is resolved. **Action:** Retry the request after allowing some time for the condition to clear.
0C	1C	**Equate Symbol:** IXCMSGORSNMSGPENDINGMUSTQUEUE **Meaning:** Environmental error. Messages pending; timeout not specified. The system could not initiate the send for this message because there are other message-out requests already pending that must be initiated first. **Action:** • Try sending the message again after allowing some time for the condition to clear. • Invoke IXCMSGO again specifying a nonzero TIMEOUT value so that XCF will queue the message and resend it automatically when the condition clears. If the condition reoccurs when a nonzero TIMEOUT value is specified, XCF will either: – Accept the message with return code X'4', or – Reject the message with a different reason code with return code X'C'.

Table 12. Return and Reason Codes for the IXCMSGO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0C	20	**Equate Symbol:** IXCMSGORSNDUALFULL **Meaning:** Environmental error. DU-AL is full. The Dispatchable Unit Access List (DU-AL) is full. Applies only when MSGACCESS=ASYNC is specified. **Action:** Retry the request after removing at least one space from the DU-AL.
0C	24	**Equate Symbol:** IXCMSGORSNDUALNOSTORAGE **Meaning:** Environmental error. Unable to obtain DU-AL storage. The system could not obtain storage to create the Dispatchable Unit Access List needed to process the request. Applies only when MSGACCESS=ASYNC is specified. **Action:** Retry the request after allowing some time for the condition to clear.
0C	28	**Equate Symbol:** IXCMSGORSNDUALNOTSUITABLE **Meaning:** Environmental error. DU-AL is unsuitable. The Dispatchable Unit Access List (DU-AL) is unsuitable. The DU=AL is too large or provided access to a subspace at some point during its existence. Applies only when MSGACCESS=ASYNC is specified. **Action:** Retry the request under some other work unit.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return and reason code and supply it to the appropriate IBM support personnel.

Example

Operation: Send a message to another active member of the XCF group. You can obtain the token of the calling member and the target member from the member information field that IXCQUERY returns. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCMSGO MEMTOKEN=TOKEN1,MSGBUF=BUFFER1,MSGLEN=LENMSG,         X
               TARGET=TOKEN2,RETCODE=RETURN,RSNCODE=REASON,MF=S

TOKEN1   DS    CL8                      MEMBER TOKEN OF MEMBER SENDING X
                                        THE MESSAGE
TOKEN2   DS    CL8                      MEMBER TOKEN OF MEMBER TO      X
                                        RECEIVE THE MESSAGE
RETURN   DS    1F                       RETURN CODE
REASON   DS    1F                       REASON CODE
BUFFER1  DC    CL256'THIS IS A TEST'    MESSAGE TO BE SENT
LENMSG   DC    F'256'                   LENGTH OF THE MESSAGE
```

You can obtain the member tokens from the QUAMTOKN field in the area returned by IXCJOIN or IXCQUERY,

IXCQUERY — Obtain XCF Information

Description

The IXCQUERY macro allows any authorized caller to request information about the resources the cross-system coupling facility (XCF) manages. The REQINFO parameter determines whether the information is about XCF groups, systems in the sysplex, the sysplex itself, coupling facility resources, or information related to the automatic restart manager.

- REQINFO=GROUP returns information about groups and members defined to XCF. The information can be about all groups, a specific group, or a single member of a group. If you issue IXCQUERY without any parameters, you receive general information about each group defined to XCF.
- REQINFO=SYSPLEX returns information about each system in the sysplex.
- REQINFO=CF returns information about coupling facilities defined in the CFRM active policy. The information can be about all coupling facilitiew or a specific coupling facility.
- REQINFO=CF_ALLDATA returns information about all coupling facilities.
- REQINFO=STR returns information about coupling facility structures defined in the CFRM active policy. The information can be about all structures or a specific structure.
- REQINFO=STR_ALLDATA returns information about all coupling facility structures.
- REQINFO=COUPLE returns information about the sysplex. The scope of the output depends on the other parameters specified.
- REQINFO=ARMSTATUS returns information about active elements. The scope of the output depends on the other parameters specified.
- REQINFO=ARMS_ALLDATA returns information about all active elements.
- REQINFO=FEATURES returns information about the XCF and XES software features installed on the system from which the request is made.

When you use REQINFO with all options except COUPLE and FEATURES, use the ANSAREA parameter to tell XCF where to return the information, and ANSLEN to tell XCF the length of the answer area. REQINFO=COUPLE and REQINFO=FEATURES generate an inline expansion.

Sections in the IXCYQUAA mapping macro provide the format for the data:

- QUAHDR maps the offset and length of the other record types.
- QUAGRP maps the group record.
- QUAMEM maps the member record.
- QUASYS and QUASYS1 map system information records.
- QUACF and QUACF1 map coupling facility records.
- QUACFSC and QUACFSC1 map system connectivity to coupling facility records.
- QUACFSTR and QUACFSTR1 map information about coupling facility structures allocated in a coupling facility.
- QUASTR and QUASTR1 map the coupling facility structure record.
- QUASTRPL and QUASTRPL1 map the coupling facility structure preference list records.
- QUASTRXL and QUASTRXL1 map the coupling facility structure exclusion list records.
- QUASTRCF and QUASTRCF1 map information about the coupling facility containing a coupling facility structure.
- QUASTRUSER and QUASTRUSER1 map the connectors to coupling facility structure records.
- QUASTRSYS maps information about system participation in a system-managed process for a coupling facility structure.
- QUAARMS maps information about automatic restart manager elements.
- QUREQFEATURES maps information about software features installed on a system.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task The following request types allow both task and SRB mode: • REQINFO=GROUP,REQTYPE=IMMEDIATE • REQINFO=SYSPLEX • REQINFO=COUPLE • REQINFO=FEATURES
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts. The request types REQINFO=COUPLE and REQINFO=FEATURES support disablement.
Locks:	No locks held. The request types REQINFO=COUPLE and REQINFO=FEATURES allow any lock to be held.
Control parameters:	Control parameters must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL). IXCQUERY with REQINFO=COUPLE and with REQINFO=FEATURES generates an inline expansion and does not require a control parameter list.

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before IXCQUERY. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

If you use REQINFO= any option except COUPLE, include the IXCYQUAA macro in the code to map the data returned in ANSAREA or FEATAREA.

If you use REQINFO=CF, REQINFO=CF_ALLDATA, STR, or STR_ALLDATA, you need to include the IXLYNDE macro in the code to map the hardware identification information associated with a coupling facility.

Restrictions

When you issue IXCQUERY REQINFO=GROUP, REQTYPE=DEFER (the default), XCF suspends the task while it serializes and accesses the most recent data.

The following request types are the only ones which allow enabled unlocked task (EUT) FRRs to be established:
- REQINFO=CF
- REQINFO=CF_ALLDATA
- REQINFO=STR
- REQINFO=STR_ALLDATA

Note: When issuing one of the above REQINFO types, XCF suspends the task while it serializes and accesses the most recent data. For all other request types, the caller cannot have any EUT FRRs established.

You cannot code REQINFO=COUPLE and REQINFO=FEATURES on the execute form of IXCQUERY.

Input Register Information

Before issuing the IXCQUERY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXCQUERY macro, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code, if applicable, and if GPR15 return code is nonzero
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller of the IXCQUERY macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro, and restore them after the system returns control

IXCQUERY with REQINFO=COUPLE and with REQINFO=FEATURES generates an inline expansion; therefore, it does not use a control parameter list, does not produce a return code, and does not preserve AR/GPRs 0, 1, 14, or 15 across the expansion.

Performance Implications

REQTYPE=DEFER (the default), requires that XCF suspend the task while it serializes and accesses the most recent data for the following REQINFO types:
- REQINFO=GROUP
- REQINFO=CF
- REQINFO=CF_ALLDATA
- REQINFO=STR
- REQINFO=STR_ALLDATA

IXCQUERY processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXCQUERY Version Support

The IXCQUERY macro supports versions in the range of 0 - 2.
- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXCQUERY macro.
- The following keywords and functions are supported by all versions starting with version 1 and higher of the IXCQUERY macro.

ELEMENT	REQINFO=ARMSTATUS
JOBNAME	RESTARTGRP
LOCAL_ONLY	SYSNAME
REQINFO=ARMS_ALLDATA	

- The following keyword is supported by all versions starting with version 2 and higher of the IXCQUERY macro.

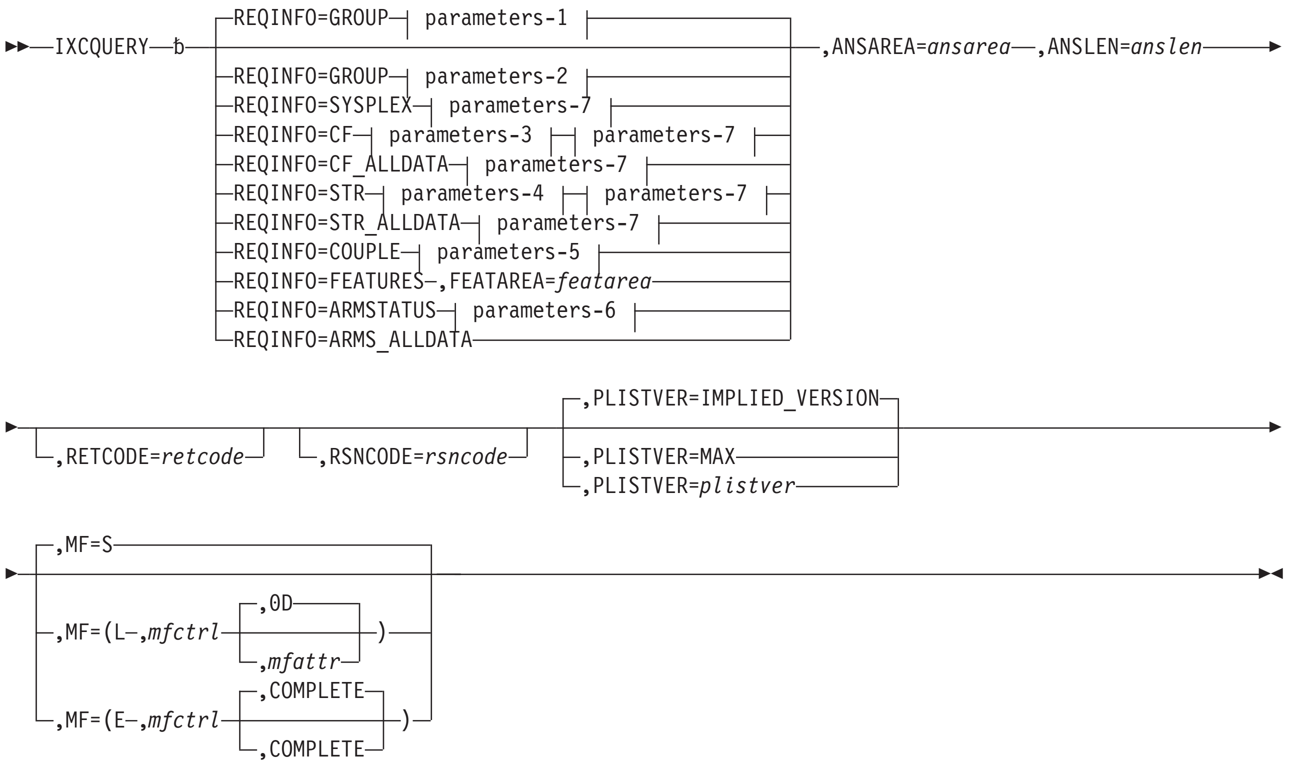
QUAALEVEL

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

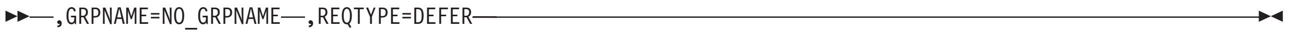
Syntax Diagram

The syntax of the IXCQUERY macro is as follows:

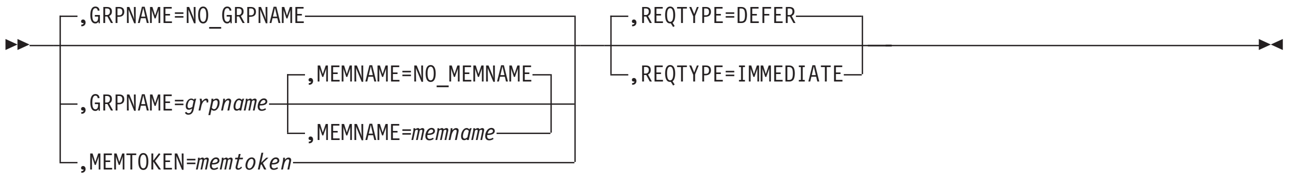
main diagram



parameters-1



parameters-2



parameters-3

```
     ┌─,CFNAME=NO_CFNAME─┐
►►───┤                   ├──────────────────────────────────────────────►◄
     └─,CFNAME=cfname────┘
```

parameters-4

```
     ┌─,STRNAME=NO_STRNAME─┐
►►───┤                     ├────────────────────────────────────────────►◄
     └─,STRNAME=strname────┘
```

parameters-5

```
►►───┬─,MAXSYS=maxsys─────────┬─────────────────────────────────────────►◄
     ├─,CURRMAXSYS=currmaxsys─┤
     ├─,LOCAL=local───────────┤
     ├─,MONOPLEX=monoplex─────┤
     ├─,SYSPLEXID=sysplexid───┤
     ├─,PLEXNAME=plexname─────┤
     ├─,CFLEVEL=cflevel───────┤
     └─,ND=nd─────────────────┘
```

parameters-6

```
►►───┬─,ELEMENT=element───────────────────────────────┬─────────────────►◄
     │                         ┌─,LOCAL_ONLY=NO──┐    │
     ├─,RESTARTGRP=restartgrp──┤                 ├────┤
     │                         └─,LOCAL_ONLY=YES─┘    │
     │                   ┌─,LOCAL_ONLY=NO──┐          │
     ├─,JOBNAME=jobname──┤                 ├──────────┤
     │                   └─,LOCAL_ONLY=YES─┘          │
     └─,SYSNAME=sysname───────────────────────────────┘
```

parameters-7

```
     ┌─,QUAALEVEL=0─────────┐
►►───┤                      ├───────────────────────────────────────────►◄
     └─,QUAALEVEL=quaalevel─┘
```

Note: REQINFO=FEATURES and REQINFO=COUPLE cannot be coded with the ANSAREA and ANSLEN keywords.

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ANSAREA=*ansarea*

Use this output parameter to specify the storage area where IXCQUERY returns the requested information. The IXCYQUAA macro provides the format of the information area. The area can be in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. Use the ANSLEN parameter to specify the length of the area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the output area where IXCQUERY is to return the requested information.

,ANSLEN=*anslen*

Use this input parameter to specify the length in bytes of the area that you provided on the ANSAREA parameter. You can estimate the length by consulting the data structures mapped by the IXCYQUAA macro. If you do not provide enough space, XCF lets you know how much space you should have provided.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword storage area which contains the length in bytes.

,CFLEVEL=*cflevel*

Use this output parameter to specify a storage area to contain the the maximum coupling facility operational level supported by the MVS operating system where the IXCQUERY macro was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword output area to contain the coupling facility level.

,CFNAME=NO_CFNAME
,CFNAME=*cfname*

Use this input parameter to specify the name of the coupling facility for which query data is to be returned. The name must be eight characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, $, @, #, and underscore (_). The name must begin with an uppercase alphabetic character.

CFNAME is an optional parameter. If CFNAME is not specified, or if CFNAME=NO_CFNAME is specified, then general information about all coupling facilities in the CFRM active policy is returned.

If a specific CFNAME is requested, then detailed information about that coupling facility is returned if the specified facility is in the active policy.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character input area which contains the name of the coupling facility.

,CURRMAXSYS=*currmaxsys*

Use this output parameter to specify a storage area to contain the maximum number of systems that can participate in the sysplex. When control returns from IXCQUERY processing, this area contains the current maximum number of systems that could participate in the sysplex given its current environment and restrictions. The value can range from 1 to the value returned for MAXSYS.

- If you specified either PLEXCFG=LOCAL or PLEXCFG=MONOPLEX in IEASYSxx, then the value of *currmaxsys* is 1.
- If you specified a different configuration with PLEXCFG, then this area contains the maximum number of systems that could appear within a sysplex.

Note: The value can change dynamically over time without a system or sysplex re-IPL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword storage area to contain the value of the current maximum number of systems.

,ELEMENT=*element*

Use this input parameter to specify the name of the element for which data is to be returned. The element name must be 16 characters long, padded on the right with blanks.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the element.

,FEATAREA=*featarea*

Use this output parameter to specify a storage area in which the system is to return information about the XCF and XES software features installed on this system.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte field to contain the XCF and XES software features as mapped by QUREQFEATURES in IXCYQUAA.

,GRPNAME=NO_GRPNAME
,GRPNAME=*grpname*

Use this input parameter to specify the name of the group for which query data is to be returned. The name must be eight characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, $, @, and #. The record returned for each member of the specified group includes the state of the member, its member token, the name of the system on which the member was last active, and all user state fields.

If GRPNAME is not specified, or if GRPNAME=NO_GRPNAME is specified, information about all groups is returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character input field which contains the name of the group.

,JOBNAME=*jobname*

Use this input parameter to specify the name of the batch job or started task for which data is to be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the jobname.

,LOCAL=*local*

Use this output parameter to specify a storage area that you use to test if the sysplex is in XCF-local mode. If the system is in XCF-local mode, then XCF returns the value X'01' in the storage area; otherwise XCF returns the value X'00' in the storage area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit output area to contain the local mode indicator.

,LOCAL_ONLY=NO
,LOCAL_ONLY=YES

Use this input parameter to specify the scope of the data for which information is to be provided.

NO

This indicates that information is to returned for elements running on any system in the sysplex that satisfy this request.

YES

This indicates that information is to be returned for all elements running on the current system.

,MAXSYS=*maxsys*

Use this output parameter to specify a storage area to contain the maximum number of systems that can participate in the sysplex, based on the MVS system level of the system on which the IXCQUERY request is issued and the sysplex configuration. When control returns from IXCQUERY:

- If you specified either PLEXCFG=LOCAL or PLEXCFG=MONOPLEX in IEASYSxx, then the value of *maxsys* is 1.
- If you specified a different configuration with PLEXCFG, then this area contains the maximum number of systems that could appear within a sysplex.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword storage area to contain the value of the maximum number of systems.

,MEMNAME=NO_MEMNAME
,MEMNAME=*memname*

Use this input parameter to specify the name of the specific member of the group for which data is to be returned. The name must be 16 characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, $, @, and #.

If MEMNAME is not specified, or if MEMNAME=NO_MEMNAME is specified, data is returned for all members of the group.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character input field which contains the name of the member.

,MEMTOKEN=*memtoken*

Use this input parameter to specify the member token of the specific member for which data is to be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte input variable that contains the member token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MONOPLEX=*monoplex*

Use this output parameter to specify a storage area that you use to test if the sysplex is in monoplex mode. If the system is in monoplex mode, then XCF returns the value X'01' in the storage area; otherwise XCF returns the value X'00' in the storage area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit output area to contain the monoplex mode indicator.

,ND=*nd*

Use this output parameter to specify a storage area to contain the node descriptor of the MVS system where the IXCQUERY macro was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte field to contain the node descriptor.

,PLEXNAME=*plexname*
Use this output parameter to specify a storage area to contain the name of the sysplex. When control returns from IXCQUERY processing, this area contains the name of the sysplex in which this system is participating.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte output area to contain the sysplex name.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXCQUERY Version Support" on page 173 for a description of the options available with PLISTVER.

,QUAALEVEL=0
,QUAALEVEL=*quaalevel*
Use this input parameter to specify the level of the IXCYQUAA record mappings that the system returns in the answer area. Valid values are 0 and 1.

- A value of 0 indicates that base level IXCYQUAA records will be returned.
- A value of 1 indicates that level-1 IXCYQUAA records will be returned. A level-1 IXCYQUAA record is identified by the final "1" in its mapping name. For example, QUACF1, QUASTR1, and QUASTRCF1 are some of the level-1 mapping macros available. To request a level-1 IXCYQUAA record, use version 2 of the IXCQUERY macro by coding QUAALEVEL=1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the one-byte area containing the level of the IXCYQUAA record mappings.

,REQINFO=GROUP
,REQINFO=SYSPLEX
,REQINFO=CF
,REQINFO=CF_ALLDATA
,REQINFO=STR
,REQINFO=STR_ALLDATA
,REQINFO=COUPLE
,REQINFO=FEATURES
,REQINFO=ARMSTATUS
,REQINFO=ARMS_ALLDATA
Use this input parameter to specify the type of information that is requested:

- REQINFO=GROUP specifies that IXCQUERY is to return data about all groups, a specific group, or a single member of a specific group.
  - When requesting information about all groups (IXCQUERY REQINFO=GROUP), the data returned includes a header record, mapped by QUAHDR, followed by one record for each group, mapped by QUAGRP.
  - When requesting information about all members of a specified group (IXCQUERY REQINFO=GROUP,GRPNAME=group_us;name, the data returned includes a header record, mapped by qUAHDR, followed by one record for each member of the specified group, mapped by QUAMEM.
  - When requesting information about a specific member of a group (IXCQUERY REQINFO=GROUP,GRPNAME=group_name,MEMNAME=member_name or IXCQUERY REQINFO=GROUP,MEMTOKEN=member_token), the data returned includes a header record, mapped by QUAHDR, followed by a record for the specified member, mapped by QUAMEM.

  The data includes a header and one record for each group requested mapped by QUAHDR, QUAGRP, and QUAMEM.
- REQINFO=SYSPLEX specifies that IXCQUERY is to return data about all systems in the sysplex. The data includes a header record, mapped by QUAHDR, and one record for each system in the

sysplex, mapped by QUASYS (QUASYS1 when QUAALEVEL=1). The record includes the system name, operator notification interval, failure detection interval, and system status.

- REQINFO=CF specifies that IXCQUERY is to return general information about all coupling facilities or detailed information about a named coupling facility. General information includes one header record, mapped by QUAHDR, and one record for each coupling facility defined in the CFRM active policy, mapped by QUACF (QUACF1 when QUAALEVEL=1). The information includes the name and ID of the coupling facility, the size of the dump space, the number of systems connected, the number of structures in the coupling facility that cannot be added to the policy, and the name and node descriptor of the coupling facility (as mapped by IXLYNDE).

  Detailed information includes one header record, mapped by QUAHDR, and information mapped by QUACFSC, QUACF, QUACFSTR, (QUACFSC1, QUACF1, and QUACFSTR1 when QUAALEVEL=1). The detailed information includes names of the systems connected to the coupling facility and names of structures and their allocation status.
- REQINFO=CF_ALLDATA specifies that IXCQUERY is to return detailed information about all coupling facilities defined in the active CFRM policy. The answer area will contain the QUAHDR, the QUACF, and the QUACFSC and QUACFSTR for each coupling facility. When QUAALEVEL=1, the answer area includes QUAHDR, QUACF1, QUACFSC1, and QUACFSTR1 for each coupling facility.
- REQINFO=STR specifies that IXCQUERY is to return general information about all coupling facility structures in the CFRM active policy or detailed information about a named coupling facility structure. General information includes a header record, mapped by QUAHDR, and one record for each coupling facility structure in the CFRM active policy, mapped by QUASTR (QUASTR1 when QUAALEVEL=1). The information includes the name of the coupling facility structure, the size, indicators as to the state of the coupling facility structure, the number of associated preference list records, the number of associated exclusion list records, the number of allocated structures with this name, and the number of connectors to the structure.

  Detailed information includes a header record, mapped by QUAHDR, and information mapped by QUASTR, QUASTRPL, QUASTRXL, QUASTRCF, QUASTRUSER, and QUASTRSYS, (QUASTR1, QUASTRPL1, QUASTRXL1, QUASTRCF1, QUASTRUSER1, and QUASTRSYS when QUAALEVEL=1). The information includes names of the coupling facilities in the structure’s preference list, names of the structures in the structure’s exclusion list, names of the coupling facilities where the structure is allocated, and detailed information about each connector to the specified structure.
- REQINFO=STR_ALLDATA specifies that IXCQUERY is to return detailed information about all coupling facility structures in the CFRM active policy. The answer area will contain the QUAHDR, the QUASTR, and the QUASTRPL, QUASTRXL, QUASTRCF, QUASTRUSER, and QUASTRSYS for each structure. When QUAALEVEL=1, the answer area includes QUAHDR, QUASTR1, QUASTRPL1, QUASTRXL1, QUASTRCF1, QUASTRUSER1, and QUASTRSYS for each structure.
- REQINFO=COUPLE specifies that IXCQUERY is to return information about the sysplex. The scope of the information depends on the other parameters specified. IXCQUERY with REQINFO=COUPLE:
  - Generates an inline expansion
  - Does not require a control parameter list
  - Does not produce a return code,
  - Does not preserve registers 0, 1, 14, or 15 across the expansion.

  Note: You cannot code REQINFO=COUPLE on the execute form of IXCQUERY.
- REQINFO=FEATURES specifies that IXCQUERY is to return information about the XCF and XES software features installed on the system from which the request is made. The system returns a bitstring mapped by QUREQFEATURES in IXCYQUAA.

  IXCQUERY with REQINFO=FEATURES:
  - Generates an inline expansion
  - Does not require a control parameter list
  - Does not produce a return code,
  - Does not preserve registers 0, 1, 14, or 15 across the expansion.

Note: You cannot code REQINFO=FEATURES on the execute form of IXCQUERY.

- REQINFO=ARMSTATUS specifies that IXCQUERY is to return information about elements that are registered with the automatic restart manager. The scope of the information depends on the other parameters specified.

  The information returned is mapped by the IXCYQUAA macro in the QUAARMS section. This information includes the name of the element, the status of the element, and the number of times the element has been restarted.
- REQINFO=ARMS_ALLDATA specifies that IXCQUERY is to return information about all elements that are registered with the automatic restart manager.

  The information returned is mapped by the IXCYQUAA macro in the QUAARMS section. This information includes the name of the elements, the status of the elements, and the number of times the elements have been restarted.

For more information about the REQINFO option, see *z/OS MVS Programming: Sysplex Services Guide*.

,REQTYPE=DEFER
,REQTYPE=IMMEDIATE

Use this input parameter to indicate whether you want the most current group and member data.
- Use **DEFER** to indicate that you are requesting the most current group and member data. The system suspends your work unit while the data requested is serialized and accessed.
- Use **IMMEDIATE** to indicate that you are requesting the in-storage version of the group and member data. The system does not suspend your work unit while the data requested is accessed.

,RESTARTGRP=*restartgrp*

Use this input parameter to specify the name of the restart group for which data is to be returned. The restart group name must be 16 characters long, padded on the right with blanks. Data on all elements in the group will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the restart group.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the return code.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the reason code.

,STRNAME=NO_STRNAME
,STRNAME=*strname*

Use this input parameter to specify the name of the coupling facility structure for which query data is to be returned. The name must be 16 characters long, padded on the right with blanks if necessary; the valid characters are A-Z, 0-9, $, @, #, and underscore (_). The name must begin with an uppercase alphabetic character. IBM structure names begin with SYS or the letters A-I.

STRNAME is an optional parameter. If STRNAME is not specified, or if STRNAME=NO_STRNAME is specified, then general information about all coupling facility structures in the CFRM active policy is returned.

If a specific STRNAME is requested, then detailed information about that coupling facility structure is returned if the specified structure is defined in the CFRM active policy.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character input area to contain the structure name.

,SYSNAME=*sysname*
Use this input parameter to specify the name of the system for which data is to be returned. Information about elements currently running on the specified system **and** elements that originally registered on the specified system will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the system name.

,SYSPLEXID=*sysplexid*
Use this output parameter to specify a storage area to contain the sysplex identifier. When control returns from IXCQUERY processing, this area contains the unique sysplex identifier, or token, that identifies the particular sysplex. The token is established when a sysplex is initialized and exists as long as the sysplex exists. (A sysplex is initialized when the first system IPLs into the sysplex.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-bit output area to contain the sysplex identifier.

Return and Reason Codes

When the IXCQUERY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code if applicable.

Mapping macro IXCYQUAA provides equate symbols for the reason codes. The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 13. Return and Reason Codes for the IXCQUERY Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** IXCQUERY completed successfully and returned the requested information. **Action:** None.
4	00000004	**Equate Symbol:** QUAARSNRECORDSREMAIN **Meaning:** Program error. IXCQUERY completed successfully and provided some data; however, ANSAREA is too small to contain all the requested data. **Action:** Take one or more of the following actions: • If the required information is contained in the ANSAREA, no further action needs to be taken. • If the required information is not contained in the ANSAREA, obtain a larger ANSAREA and reissue IXCQUERY. The QUAHTLEN field contains the ANSAREA size that is required to contain all of the information requested. However, because IXCQUERY returns only a snapshot of the current environment, it is possible that the QUAHTLEN may be too small on the next invocation. • Ensure that you specified the correct length for the answer area.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	00000004	**Equate Symbol:** QUAARSNGROUPNOTFOUND **Meaning:** Program error. The group name specified is not defined to XCF. **Action:** Take one or more of the following actions: • Ensure that the correct group name, group name address, and ALET (if appropriate) were specified. • Correct the group name and retry the request. • Any further action depends on your application. Some type of recovery action might need to be taken. However, if the group no longer exists, no action is required.
8	00000008	**Equate Symbol:** QUAARSNREQINFONOTVALID **Meaning:** Program error. The REQINFO information is not valid. **Action:** Take one or more of the following actions: • Ensure that GROUP, SYSPLEX, COUPLE, CF, CF_ALLDATA, STR_ALLDATA, STR, ARMSTATUS, or ARMS_ALLDATA was specified for REQINFO. • If REQINFO=GROUP was specified, ensure that GRPNAME or MEMNAME was specified. • Ensure that the parameter list was not inadvertently overlaid. • Ensure that the correct parameter list and ALET were specified. • If your program is running in AR ASC mode, ensure that SYSSTATE ASCENV=AR was specified.
8	0000000C	**Equate Symbol:** QUAARSNREQTYPEINCOR **Meaning:** Program error. The caller specified the REQTYPE control parameter incorrectly. **Action:** Take one or more of the following actions: • Ensure that a REQTYPE of DEFER or IMMEDIATE was specified. • Ensure that the parameter list was not inadvertently overlaid. • Ensure that the correct parameter list address and ALET were specified. • If your program is running in AR mode, ensure that SYSSTATE ASCENV=AR was specified.
8	00000010	**Equate Symbol:** QUAARSNMEMBERNOTFOUND **Meaning:** Program error. The member name specified is not defined within the specified group, or the MEMTOKEN specified was not defined to XCF. **Action:** The action you take depends on your application. If you have reason to believe that the member is defined, ensure that: • The parameter list was not inadvertently overlaid. • The correct parameter list address and ALET were specified. • SYSSTATE ASCENV=AR was specified if your program is running in AR mode.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	00000014	**Equate Symbol:** QUAARSNANSAREATOOSMALL **Meaning:** Program error. The length the caller specified on ANSLEN is too small to contain even the header. **Action:** Take one or more of the following actions: • Provide an answer area that is large enough to contain at least the header information as mapped by the field QUAHDR of the query answer area mapping. • Ensure that you specified the correct length for the answer area. • Ensure that the parameter list was not inadvertently overlaid. • Ensure that you specified the correct answer area address and ALET. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR
8	00000018	**Equate Symbol:** QUAARSNANSAREANOACCESS **Meaning:** Program error. XCF cannot access the area specified by ANSAREA. **Action:** Take one or more of the following actions: • Ensure that the correct ANSAREA address, ALET, and ANSLEN were specified. • Ensure that the parameter list storage area was not inadvertently freed by your program. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCQUERY macro.
8	0000001C	**Equate Symbol:** QUAARSNANSALETNOTVALID **Meaning:** Program error. The ALET that qualifies the address of the ANSAREA is neither zero nor is it associated with a valid public entry on the DU-AL. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • If your program is running in AR mode, you specified SYSSTATE ASCENV=AR before issuing the IXCQUERY macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET).
8	00000020	**Equate Symbol:** QUAARSNCFNOTFOUND **Meaning:** Program error. The coupling facility name specified is not defined in the CFRM active policy. **Action:** Take one or more of the following actions: • Ensure that the correct coupling facility name was specified. • Correct the coupling facility name and retry the request. • Any further action depends on your application. Some type of recovery action might need to be taken. However, if the coupling facility is no longer defined in the CFRM active policy, no action is required.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	00000024	**Equate Symbol:** QUAARSNSTRNOTFOUND **Meaning:** Program error. The structure name specified is not defined in the CFRM active policy. **Action:** Take one or more of the following actions: • Ensure that the correct coupling facility structure name was specified. • Correct the coupling facility structure name and retry the request. • Any further action depends on your application. Some type of recovery action might need to be taken. However, if the coupling facility is no longer defined in the CFRM active policy, no action is required.
8	00000028	**Equate Symbol:** QUAARSNARMNAMENOTFOUND **Meaning:** Program error. The job, element, system, or restart group name specified is not known to the automatic restart manager. **Action:** The action you take depends on your application. If you have reason to believe that the name is correct ensure that: • The parameter list was not inadvertently overlaid. • The correct parameter list address and ALET were specified. • SYSSTATE ASCENV=AR was specified if your program is running in AR mode. • The correct name was specified.
8	00000034	**Equate Symbol:** QUAARSNAMODE24 **Meaning:** Program error. The IXCQUERY macro was issued in 24-bit addressing mode. **Action:** IXCQUERY runs in 31-bit addressing mode. Correct your program so that it calls IXCQUERY while in 31-bit addressing mode.
8	00000040	**Equate Symbol:** QUAARSNBADPLISTRSVD **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
8	000000A0	**Equate Symbol:** IXCARMINVR0 **Meaning:** System error. **Action:** Make sure your program was assembled with the correct macro library for the release of MVS your program is running on. If the macro version is correct, retry the request at least once.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	000000A4	**Equate Symbol:** IXCARMR0TYPECONFL **Meaning:** System error. **Action:** Verify that: • The parameter list was not inadvertently overlaid. • The parameter list was initialized. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • If IXCQUERY was called in AR mode: – The SYSSTATE ASCENV=AR macro was issued prior to this macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – This area is either in the primary address space or in an address/data space that is addressable through a public entry on the caller's DU-AL. If everything is verified, retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.
8	00000100	**Equate Symbol:** QUAARSNPLISTALETNOTVALID **Meaning:** Program error. The ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • Your program is not intended to run in primary ASC mode. • You specified SYSSTATE ASCENV=AR before issuing the IXCQUERY macro. • The ALET for the parameter list is a valid public entry on the DU-AL or is zero (primary address space ALET).
8	00000104	**Equate Symbol:** QUAARSNVERSIONNOTVALID **Meaning:** Program or environmental error. The version number in the control parameter list is not valid. Your program might have inadvertently written over an area in the control parameter list. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
8	00000108	**Equate Symbol:** QUAARSNFUNCCODENOTVALID **Meaning:** Program error or environmental error. The function code in the control parameter list is not valid. Your program might have inadvertently written over an area in the control parameter list. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	0000010C	**Equate Symbol:** QUAARSNPLISTNOACCESS **Meaning:** Program error or environmental error. XCF could not access the control parameter list. **Action:** Check to see if your program inadvertently freed or overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
8	00000118	**Equate Symbol:** QUAARSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCQUERY while in task mode.
8	0000011C	**Equate Symbol:** QUAARSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCQUERY while it is enabled.
8	00000120	**Equate Symbol:** QUAARSNHASLOCK **Meaning:** Program error. The caller is holding a lock. **Action:** Correct your program so that it issues IXCQUERY when it is not holding a lock.
8	00000124	**Equate Symbol:** QUAARSNHASEUTFRR **Meaning:** Program error. The caller has an EUT FRR established. **Action:** Correct your program so that it issues IXCQUERY when it does not have an EUT FRR established.
8	00000128	**Equate Symbol:** QUAARSNQUAALEVELNOTVALID **Meaning:** Program error. The caller specified a value for QUAALEVEL that is not valid. **Action:** Correct your program so that it specifies a valid value for QUAALEVEL.
C	00000004	**Equate Symbol:** QUAARSNDSPSERVFAIL **Meaning:** XCF was unable to create a data space for an IXCQUERY request. This reason code applies only to IXCQUERY requests REQINFO=CF, CF_ALLDATA, STR, or STR_ALLDATA. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	00000008	**Equate Symbol:** QUAARSNALESERVFAIL **Meaning:** XCF was unable to associate a data space to an address space on behalf of an IXCQUERY request. This reason code applies only to IXCQUERY requests REQINFO=CF, CF_ALLDATA, STR, or STR_ALLDATA. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.
C	00000018	**Equate Symbol:** QUAARSNTASKABENDED **Meaning:** While the issuing task was suspended for XCF processing, the system abended the task (that is, another unit of work attempted to abnormally terminate this task). No data was returned in the ANSAREA. This reason code applies only to REQINFO=GROUP,REQTYPE=DEFER requests. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.
C	00000144	**Equate Symbol:** QUAARSNNOCFRMDSN **Meaning:** Environmental error. The CFRM active policy could not be read because the couple data set supporting TYPE(CFRM) is not accessible to this system. This reason code applies only to IXCQUERY requests REQINFO=CF, CF_ALLDATA, STR, or STR_ALLDATA. **Action:** Ensure that the request is issued from a system with access to a CFRM couple data set.
C	00000154	**Equate Symbol:** QUAARSNNOCFRMPOL **Meaning:** A CFRM policy has not been activated. This reason code applies only to IXCQUERY requests REQINFO=CF, CF_ALLDATA, STR, or STR_ALLDATA. **Action:** Ensure that a CFRM policy is active.
C	0000015C	**Equate Symbol:** QUAARSNFAILCFRMREAD **Meaning:** The CFRM active policy could not be read because the couple data set supporting TYPE(CFRM) is in error. This reason code applies only to IXCQUERY requests REQINFO=CF, CF_ALLDATA, STR, or STR_ALLDATA. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.

Table 13. Return and Reason Codes for the IXCQUERY Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	00000160	**Equate Symbol:** QUAARSNNOARMDSN **Meaning:** Environmental error. The automatic restart management couple data set could not be read because the couple data set supporting TYPE(ARM) is not accessible to this system. This reason code applies only to IXCQUERY requests REQINFO=ARMSTATUS and ARMS_ALLDATA. **Action:** If this reason code is not expected, contact the system programmer or issue a message to the operator to see if the automatic restart management couple data set can be brought online for this system. Issue this IXCQUERY request from a system that is connected to the automatic restart management couple data set.
C	00000164	**Equate Symbol:** QUAARSNFAILARMREAD **Meaning:** Environmental error. The automatic restart manager active policy could not be read because the couple data set supporting TYPE(ARM) is in error. This reason code applies only to IXCQUERY requests REQINFO=ARMSTATUS, and ARMS_ALLDATA. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.
10	xxxxxxxx	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.

Example

Operation: Request the most recent information about a specific member MEMB1 of the group GROUPA. Register 2 points to the area where XCF is to place the member information. XCF is to store the return code and reason code into the RETURN and REASON fields. The code is as follows:

```
         LA    R2,MYAREA                OBTAIN ADDRESS OF OUTPUT AREA  X
                                        FOR IXCQUERY

         IXCQUERY REQINFO=GROUP,GRPNAME=GROUPA,ANSAREA=(R2),           X
               ANSLEN=AREALEN,REQTYPE=DEFER,RETCODE=RETURN,            X
               RSNCODE=REASON,MEMNAME=MEMB1,MF=S

GROUPA   DC    CL8'GROUPA'              NAME OF GROUP FOR WHICH DATA   X
                                        IS TO BE RETURNED
MEMB1    DC    CL16'MEMB1'              NAME OF MEMBER FOR WHICH DATA  X
                                        IS TO BE RETURNED
MYAREA   DS    CL156                    OUTPUT AREA TO CONTAIN DATA    X
                                        RETURNED BY IXCQUERY
RETURN   DS    1F                       RETURN CODE
REASON   DS    1F                       REASON CODE
AREALEN  DC    F'156'                   LENGTH OF OUTPUT AREA
```

IXCQUIES — Place an XCF Member in a Quiesced State

Description

The IXCQUIES macro allows an active XCF member to place itself in a quiesced state. In a quiesced state, the member can no longer use the monitoring and signalling services of XCF. If active members have a group user-routine, XCF notifies those members that the member is quiesced. XCF delivers outstanding messages sent by the member, and discards undelivered messages that were sent to the member.

IXCQUIES requires that the member have permanent status recording established.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any HASN, any SASN. The primary address space must be the same as the primary address space of the caller of the IXCJOIN macro that placed the calling member in the active state, or the caller must be executing in the master scheduler address space.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCQUIES. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

The member must be active, with permanent status recording established.

The caller can have no enabled, unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCQUIES macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	**Contents**
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

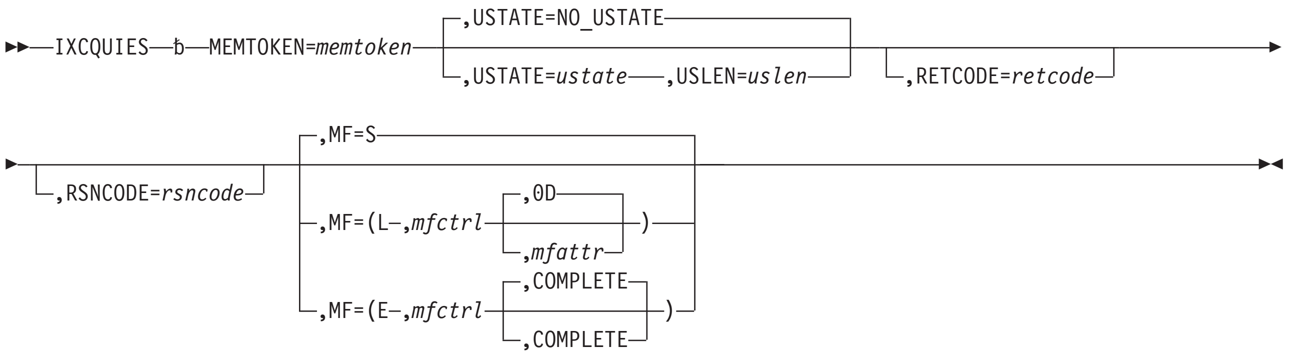
Performance Implications

None.

Syntax Diagram

The syntax of the IXCQUIES macro is as follows:

IXCQUIES diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

MEMTOKEN=*memtoken*

Use this input parameter to specify the 64-bit token of the member. XCF provided this token when it placed the member in the active state.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the quiesce requests completes.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the quiesce request completes.

,USLEN=*uslen*
Use this input parameter to specify the length in bytes of the user state data that you provide on the USTATE parameter. The length must be from 1 to 32 bytes. XCF overlays any previous value in the user state field up to the length you specify on USLEN. XCF does not pad the remainder of the user state field (up to 32 bytes) with zeros. IXCQUERY always returns the full 32 bytes, and group user-routines always receive the full 32 bytes. If you code USTATE, USLEN is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the length of the user state data (USTATE).

,USTATE=NO_USTATE
,USTATE=*ustate*
Use this input parameter to specify the area containing data that you want XCF to place in the user state field associated with the member. Use the USLEN parameter to specify the length of the user state data.

If you do not specify USTATE, or if you specify USTATE=NO_USTATE, the user state field remains unchanged.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) that contains the user state information.

ABEND Codes

None.

Return and Reason Codes

When the IXCQUIES macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 14. Return and Reason Codes for the IXCQUIES Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
00	None.	**Meaning:** IXCQUIES completed successfully; XCF places the member in a quiesced state. **Action:** None.
04	04	**Equate Symbol:** IXCQUIESRSNEXITSNOTPURGED **Meaning:** Environmental error. IXCQUIES completed successfully; XCF did not purge the group, status, or message user-routine SRBs successfully. For a group, status, or message exit, it is possible that the member specified on IXCJOIN is still running. Once an exit returns to XCF, it will not be scheduled again. XCF has attempted to purge the exits several times. This condition occurs only if there was an XCF error or the current task was asynchronously abended several times while XCF was in control to process the IXCQUIES request. **Action:** Your application should be aware that one of the exits might still be running. If this return code occurs more than once for IXCQUIES, take the following actions: • Determine if any asynchronous abends are being issued against the current task. If so, you might need to reduce their frequency. • XCF should have taken an SDUMP to record the abend or abends. If the SDUMP indicates that the error was caused by SRB-to-task percolation or application code issuing a CALLRTM macro against your task, this is probably not an XCF error. If you believe this is an XCF error, record the return and reason code, and supply it to the appropriate IBM support personnel.

Table 14. Return and Reason Codes for the IXCQUIES Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	04	**Equate Symbol:** IXCQUIESRSNNOTACTIVE **Meaning:** Program error. The member token does not identify an active member. **Action:** Ensure that the correct MEMTOKEN was specified. Further action depends on your application. If the member must be in a not-defined state, you can use the IXCQUERY service to determine what state the member is currently in. If the member is currently in a failed, quiesced, or created state, use the IXCDELET service to place the member in a not-defined state.
08	08	**Equate Symbol:** IXCQUIESRSNINAPPROPRIATEPRIMARY **Meaning:** Program error. The primary address space is neither the master scheduler address space nor the primary address space of the caller of the IXCJOIN that placed the member in the active state. **Action:** Ensure that: • The correct MEMTOKEN was specified. • Your program issues IXCQUIES only from the master scheduler address space or the address space from which the member (MEMTOKEN) joined the group.
08	0C	**Equate Symbol:** IXCQUIESRSNNOTLASTING **Meaning:** Program error. The caller specified a member token for a member that did not specify LASTING=YES on the IXCJOIN macro. To be placed in a quiesced state, a member has to request permanent status recording. **Action:** Ensure that the correct MEMTOKEN was specified. If your application requires the member to have permanent status recording, ensure that LASTING=YES is specified on the IXCJOIN that joins the member to the group.
08	10	**Equate Symbol:** IXCQUIESRSNINAPPROPRIATESYSTEM **Meaning:** Program error. The system is not the system on which the IXCJOIN for the member was issued. **Action:** Ensure that: • The correct MEMTOKEN was specified. • Your program issues IXCQUIES only for members that joined on the same system as your program.
08	40	**Equate Symbol:** IXCQUIESRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 14. Return and Reason Codes for the IXCQUIES Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	100	**Equate Symbol:** IXCQUIESRSNPLISTBADALET **Meaning:** Program error. Your program is running in AR mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCQUIES macro. • The ALET for the parameter list is on the DU-AL or is zero (primary address space ALET). • Your program is not running in primary ASC mode.
08	104	**Equate Symbol:** IXCQUIESRSNPLISTVERSIONNOTVALID **Meaning:** Program error or environmental error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCQUIESRSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.
08	10C	**Equate Symbol:** IXCQUIESRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode, ensure that: – You specified SYSSTATE ASCENV=AR before issuing the IXCQUIES macro. – The parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
08	110	**Equate Symbol:** IXCQUIESRSNUSTATEBADSTG **Meaning:** Program error. XCF could not access the USTATE value. **Action:** Take one or more of the following actions: • Ensure that the correct USTATE address was specified. • If your program is running in AR mode, ensure that: – You specified SYSSTATE ASCENV=AR before issuing the IXCQUIES macro. – The USTATE ALET is correct. • Ensure that the USTATE storage area was not inadvertently freed by your program.

Table 14. Return and Reason Codes for the IXCQUIES Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	114	**Equate Symbol:** IXCQUIESRSNUSLENBADVALUE **Meaning:** Program error. The USLEN value is less than 1 or greater than 32. **Action:** Correct the USLEN, and retry the request.
08	118	**Equate Symbol:** IXCQUIESRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCQUIES only while in task mode, and retry the request.
08	11C	**Equate Symbol:** IXCQUIESRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCQUIES only while enabled.
08	120	**Equate Symbol:** IXCQUIESRSNPRIMARYNOTHOME **Meaning:** Program error. The primary address space is not equal to the home address space. **Action:** Correct your program so that it does not use IXCQUIES while in cross memory mode. You might want to pass this restriction on to your caller when you are unsure of the environment your caller might have been in.
0C	18	**Equate Symbol:** IXCQUIESRSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended; that is, another unit of work attempted to abnormally terminate this task). The state of the IXCQUIES request is unpredictable. **Action:** Find out why this task was being abended.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request one or more times. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Operation: Place a member in the quiesced state, and put the value X'33' in the user state field. XCF is to store the return code and reason code into the RETURN and REASON fields. The code is as follows:

```
         IXCQUIES MEMTOKEN=TOKEN2,USTATE=STATE3,USLEN=LEN,             X
               RETCODE=RETURN,RSNCODE=REASON,MF=S

TOKEN2   DS    CL8                       TOKEN OF MEMBER TO BE PLACED   X
                                         IN QUIESCED STATE
RETURN   DS    1F                        RETURN CODE
REASON   DS    1F                        REASON CODE
STATE3   DC    X'33'                     USER STATE VALUE
LEN      DC    F'1'                      LENGTH OF USER STATE DATA
```

You can obtain the member token from the QUAMTOKN field in the area returned by IXCJOIN or IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXCSETUS — Update the User State Field

Description

The IXCSETUS macro updates the user state field that the cross-system coupling facility (XCF) maintains for an XCF member. The target member must be in the same XCF group as the caller and must be in one of the following states: created, active, quiesced, or failed; the caller can be the target member. The caller specifies, on the NEWUS parameter, what the new value of the target member's user state field is to be. When a change is made to the user state field, XCF broadcasts the change to those active members in the group that have group user routines.

Before it requests a change in the user state field, a caller can ask XCF to confirm the current value of the field. XCF compares the current value with a *compare value* that COMPUS identifies. This operation is similar to a *compare and swap* instruction.

- If the current value and the compare value match, XCF replaces the current value with the value that NEWUS points to.
- If the current value and the compare value do not match, XCF leaves the current value of the user state field as it is. If you specify OLDUS, XCF copies the current value to that field. The USLEN parameter determines the length of the data XCF copies to OLDUS.

If the OLDUS and COMPUS parameters both point to the same field, XCF replaces the compare value with the current value.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space of the caller of the IXCJOIN that placed the calling member in the active state.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry in the dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before IXCSETUS. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

- The caller must be a member in the active state, and the target member must be created, active, quiesced, or failed; both must be in the same XCF group.
- The caller can have no enabled, unlocked task (EUT) FRRs established.

Input Register Information

Before issuing the IXCSETUS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Understanding IXCSETUS Version Support

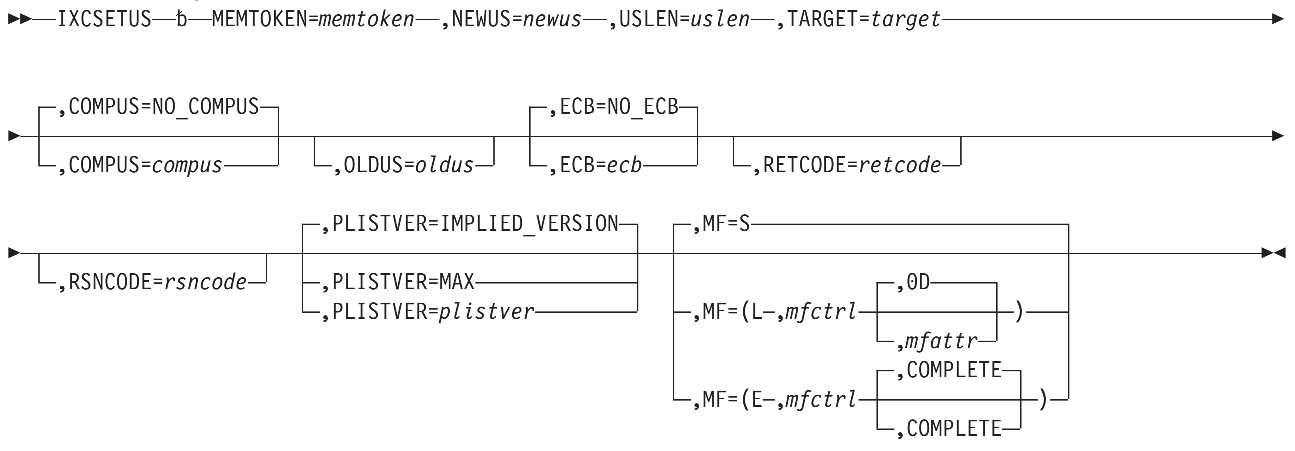
The IXCSETUS macro supports version 0 keywords and functions.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCSETUS macro is as follows:

IXCSETUS diagram



Parameter Descriptions

The parameters are explained as follows. Default values are underlined:

,COMPUS=NO_COMPUS
,COMPUS=*compus*

Use this input parameter to specify the user state compare value. If the compare value matches the current value, XCF replaces the current value with the new value. If the values do not match, XCF does not change the user state field.

If you do not want to compare the user state with a value, but would like to code this parameter, specify COMPUS=NO_COMPUS.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) to be used for comparison against the current user state field for the target.

,ECB=NO_ECB
,ECB=*ecb*

Use this input parameter to specify a 32-bit ECB that XCF is to post when IXCSETUS completes its actions. If you do not specify ECB, or if you specify ECB=NO_ECB, IXCSETUS completes synchronously. If you do specify ECB, XCF might return control before IXCSETUS completes its actions. If the return code is 0, indicating that XCF accepted the request, then issue a WAIT on the ECB. If the return code is not 0, XCF will not post the ECB.

If you specify OLDUS with ECB, then XCF updates the OLDUS field just before it posts the ECB. In this case, the OLDUS field must be in commonly addressable storage.

When XCF posts the ECB, the ECB contains the following:

- Bits 8 - 23 contain the low-order halfword of the reason code that XCF would have returned in GPR 0 if the caller had not specified ECB.
- Bits 24 - 31 contain the low-order byte of the return code that XCF would have returned in GPR 15 if the caller had not specified ECB.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB.

MEMTOKEN=*memtoken*

Use this input parameter to specify the 64-bit token of the calling member. IXCJOIN provided this token when it placed the member in the active state.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,NEWUS=*newus*
Use this input parameter to specify the value that XCF is to place in the user state field. This value is called the “new value.” If you specify COMPUS and the current value and the compare value do not match, the new value will not replace the current value of the field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) that contains the new user state information.

,OLDUS=*oldus*
Use this output parameter to specify the location where XCF is to copy the current value of the user state field. If OLDUS and COMPUS both point to the same location, XCF updates the existing compare value with the current value of the user state field. You might specify this parameter to find out why the comparison failed.

If you specify ECB with OLDUS, then XCF updates the OLDUS field just before it posts the ECB. In this case, the OLDUS field must be in commonly addressable storage, and any ALET that qualifies its address must be zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the area (with a length of USLEN) where the current value of the user state field of the target member will be placed.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXCSETUS Version Support” on page 200 for a description of the options available with PLISTVER.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when this request completes.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request completes.

,TARGET=*target*

Use this input parameter to specify the 64-bit token of the member whose user state field is to be changed or used for comparison. You can get the value of this token by using the IXCQUERY macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token of the member whose user state you are processing.

,USLEN=*uslen*

Use this input parameter to specify how many bytes of the user state field XCF is to change. XCF uses the length you specify on USLEN as the length of the COMPUS, OLDUS, and NEWUS fields:

- When XCF compares the current user state value with the value in COMPUS, XCF compares only the number of bytes you specify on USLEN.
- When XCF copies the current value of the user state field to OLDUS, XCF copies only the number of bytes you specify on USLEN.
- When XCF updates the current user state value with the value in NEWUS, XCF updates only the number of bytes you specify on USLEN.

The length must be from 1 to 32 bytes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of bytes of the user state field to process.

ABEND Codes

None.

Return and Reason Codes

When the IXCSETUS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 15. Return and Reason Codes for the IXCSETUS Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** IXCSETUS completed successfully as the caller requested. If the caller specified ECB, the processing is asynchronous; if the caller omitted ECB, the processing is synchronous. **Action:** None.

Table 15. Return and Reason Codes for the IXCSETUS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
04	04	**Equate Symbol:** IXCSETUSRSNNOCHANGEOLDEQNEW **Meaning:** Program error. XCF did not change the user state value and did not notify members. If the caller specified OLDUS, XCF returned the current user state value. COMPUS was not specified, and the new value is the same as the current value. **Action:** None required, because the current user state already matches the desired user state.
04	08	**Equate Symbol:** IXCSETUSRSNNOCHANGEOLDNECOMPUS **Meaning:** This return code is informational. COMPUS was specified, and the compare value is not equal to the current value. XCF did not change the user state field and did not notify members. If the caller specified OLDUS, XCF returned the current user state value. **Action:** None required. The user state should be updated only when the current value matches the COMPUS value. However, if COMPUS is being used to serialize user state updates with other user state update requests, you can retry the request with a new user state value based on the OLDUS, and use the OLDUS as the COMPUS value.
08	04	**Equate Symbol:** IXCSETUSRSNNOTACTIVE **Meaning:** Program error. The caller's member token does not identify an active member. **Action:** Ensure that the correct MEMTOKEN was specified. Further action depends on your application. If the member must be placed in a not-defined state, you can use the IXCQUERY service to determine what state the member is currently in. If the member is currently in a failed, quiesced, or created state, you can use the IXCDELET service to place the member in a not-defined state.
08	08	**Equate Symbol:** IXCSETUSRSNINAPPROPRIATEPRIMARY **Meaning:** Program error. The primary address space is not the same as the primary address space of the caller of the IXCJOIN that placed the calling member in the active state. **Action:** Ensure that the correct MEMTOKEN was specified. Change your program to issue IXCSETUS from the same primary address space as the primary address space of the caller of the IXCJOIN that placed the calling member in the active state.
08	0C	**Equate Symbol:** IXCSETUSRSNTARGETDIFFERENTGROUP **Meaning:** Program error. The calling member and the target member are not members of the same XCF group. **Action:** Take one or more of the following actions: • Ensure that the correct TARGET and MEMTOKEN member tokens were specified. • Correct your program so that the member updating the user state updates only the user state of members within the same group.

Table 15. Return and Reason Codes for the IXCSETUS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	10	**Equate Symbol:** IXCSETUSRSNTARGETNOTVALID **Meaning:** Program error. The target member's token is not valid. **Action:** Correct the target member token, and retry the request.
08	14	**Equate Symbol:** IXCSETUSRSNOLDUSALETNOTPRIMARY **Meaning:** Program error. The caller specified ECB and OLDUS, and is running in AR mode. However, the ALET that qualifies the address of the OLDUS field is not primary. If you specify ECB, the request completes asynchronously; therefore, the ALET for the OLDUS must be primary, and the OLDUS address must be in common storage. See the requirements for the OLDUS keyword. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. • The ALET of the OLDUS field is zero (primary address space ALET).
08	18	**Equate Symbol:** IXCSETUSRSNOLDUSBADSTGNOTCOMMON **Meaning:** Program error. The caller specified ECB and OLDUS; however, the OLDUS field is not in common storage. **Action:** Ensure that the correct OLDUS field address was used, and that the field is in common storage.
08	28	**Equate Symbol:** IXCSETUSRSNOLDUSBADALET **Meaning:** Program error. The caller specified OLDUS, but provided an inappropriate ALET. The ALET is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. • The ALET is a public entry on the DU-AL or zero (primary address space ALET). • Your program is not intended to run in primary ASC mode.

Table 15. Return and Reason Codes for the IXCSETUS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	3C	**Equate Symbol:** IXCSETUSRSNOLDUSINCOMPLETE **Meaning:** Program error. The caller specified OLDUS incorrectly. The user state might or might not have been updated. Check the two high-order bytes of this reason code fullword *xxyy*003C for the return code *xx* (either 00 or 04) and reason code *yy* the caller would have received if the caller had coded those parameters correctly. **Action:** Take one or more of the following actions: • Ensure that the correct OLDUS address was used. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. – Ensure that the OLDUS ALET is correct. • Ensure that the OLDUS storage area was not inadvertently freed by your program. • You might want to abnormally end your program or take some other action that will record the problem with your OLDUS storage area.
08	40	**Equate Symbol:** IXCSETUSRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	100	**Equate Symbol:** IXCSETUSRSNPLISTBADALET **Meaning:** Program error. Your program is running in primary ASC mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid public entry on the caller's DU-AL. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. • The ALET for the parameter list is on the DU-AL or is zero (primary address space ALET).
08	104	**Equate Symbol:** IXCSETUSRSNPLISTVERSIONNOTVALID **Meaning:** Program error. A version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	108	**Equate Symbol:** IXCSETUSRSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.

Table 15. Return and Reason Codes for the IXCSETUS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	10C	**Equate Symbol:** IXCSETUSRSNPLISTBADSTG **Meaning:** Program error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. – Ensure that the parameter list ALET is correct.
08	110	**Equate Symbol:** IXCSETUSRSNNEWUSNOTACCESSIBLE **Meaning:** Program error. XCF could not access the NEWUS value. **Action:** Take one or more of the following actions: • Ensure that the correct NEWUS address was used. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. – Ensure that the NEWUS ALET is correct.
08	114	**Equate Symbol:** IXCSETUSRSNUSLENBADVALUE **Meaning:** Program error. The length value in USLEN is less than 1 or greater than 32. **Action:** Correct the USLEN, and retry the request.
08	118	**Equate Symbol:** IXCSETUSRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCSETUS only while in task mode.
08	11C	**Equate Symbol:** IXCSETUSRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCSETUS only while enabled.
08	124	**Equate Symbol:** IXCSETUSRSNCOMPUSNOTACCESSIBLE **Meaning:** Program error. XCF could not access the COMPUS value. **Action:** Take one or more of the following actions: • Ensure that the correct COMPUS address was used. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCSETUS macro. – Ensure that the COMPUS ALET is correct.

Table 15. Return and Reason Codes for the IXCSETUS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0C	18	**Equate Symbol:** IXCSETUSRSNTASKABENDED **Meaning:** Environmental error. While the issuing task was suspended for XCF processing, the task was abended; that is, another unit of work attempted to abnormally terminate this task. The state of the IXCSETUS request is unpredictable. **Action:** Determine why this task was abended.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Operation: If the user state current value for the target member is X'22', replace that value with the value X'44', and save the current value. In the IXCSETUS macro,
- COMPUS points to the value 22.
- NEWUS points to the value 44.
- OLDUS points to the area where XCF will save the current value.

For OLDUS, the storage must be in common storage. In this example, the STORAGE OBTAIN macro returns the address of the area in register 1. The code moves the address to register 3. XCF is to store the return code and reason code into the fields RETURN and REASON. The code is as follows:

```
* ZERO ECB1
       STORAGE OBTAIN,LENGTH=LEN,SP=228  OBTAIN STORAGE FOR OLD USER   X
                                       STATE VALUE
       LR    R3,R1                     SAVE ADDRESS FOR IXCSETUS       X
                                       INVOCATION

       IXCSETUS MEMTOKEN=TOKEN1,NEWUS=STATE4,USLEN=LEN,                X
             TARGET=TOKEN2,COMPUS=STATE2,OLDUS=(R3),ECB=ECB1,          X
             RETCODE=RETURN,RSNCODE=REASON,MF=S

* TEST RETURN CODE
       WAIT  ECB=ECB1                  WAIT FOR IXCSETUS TO COMPLETE

TOKEN1   DS    CL8                       MEMBER TOKEN OF CALLER
TOKEN2   DS    CL8                       TOKEN OF MEMBER WHOSE USER    X
                                         STATE IS TO BE CHANGED
RETURN   DS    1F                        RETURN CODE
REASON   DS    1F                        REASON CODE
ECB1     DS    1F                        ECB TO BE POSTED BY XCF
STATE2   DC    X'22'                     USER STATE FOR COMPARISON
STATE4   DC    X'44'                     NEW USER STATE VALUE
LEN      DC    F'1'                      LENGTH OF ALL USER STATES
```

You can obtain the member tokens from the QUAMTOKN field in the area returned by IXCCREAT, IXCJOIN, or IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXCSYSCL — Notify the System that Cleanup Has Completed

Description

When a system leaves the sysplex, automatic restart management will wait for all the members that specified SYSCLEANUPMEM=YES on the IXCJOIN macro to issue the IXCSYSCL macro before it will perform restarts for elements that were on the failed system.

When a system leaves the sysplex, members of cross-system coupling facility (XCF) groups will be notified through their group user routine. If SYSCLEANUPMEM=YES was specified by a member of the group, IXCSYSCL must be issued to indicate when cleanup has been completed by an XCF member (specified in the MEMTOKEN parameter), for the system that left the sysplex (specified in the FAILEDSYS parameter), or that no cleanup is necessary.

Note: Automatic restart management will not wait forever for cleanup processing to be completed. After a certain amount of time, automatic restart management will proceed with restarts regardless of whether there are any outstanding IXCSYSCL macros to be issued.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry in the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCSYSCL. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

This macro must be issued from the same address space from which the IXCJOIN was issued.

Input Register Information

Before issuing the IXCSYSCL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero, GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.

15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None.

Understanding IXCSYSCL Version Support

The IXCSYSCL macro supports version 0.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXCSYSCL macro is as follows:

IXCSYSCL diagram

```
                   ┌─PLISTVER=IMPLIED_VERSION─┐
►►──IXCSYSCL──b────┼──────────────────────────┼─,MEMTOKEN=memtoken──,FAILEDSYS=failedsys──────────►
                   ├─PLISTVER=MAX─────────────┤
                   └─PLISTVER=plistver────────┘

                                                ┌─,MF=S─────────────────────────────────┐
►──┬────────────────────┬──┬────────────────────┬─┼───────────────────────────────────────┼──────────►◄
   └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘ │                   ┌─,0D──────┐        │
                                                ├─,MF=(L─,mfctrl──────┼──────────┼─)──────┤
                                                │                   └─,mfattr──┘        │
                                                │                   ┌─,COMPLETE─┐       │
                                                └─,MF=(E─,mfctrl──────┼───────────┼─)─────┘
                                                                    └─,COMPLETE─┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,FAILEDSYS=*failedsys*

Use this input parameter to specify the name of the field that contains the system token of the system for which the cleanup has been completed. This token is from the group exit parameter list (mapped by IXCYGEPL) in the field GEPLSID. The group exit was given control when a system was removed from the sysplex.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the token for the system that left the sysplex.

,MEMTOKEN=*memtoken*

Use this input parameter to specify the name of the field that contains the member token for the member issuing this macro. This member must be active on the current system, and must have issued the IXCJOIN macro with SYSCLEANUPMEM=YES specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXCSYSCL Version Support” on page 210 for a description of the options available with PLISTVER.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request is complete.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request completes.

ABEND Codes

None.

Return and Reason Codes

When the IXCSYSCL macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 16. Return and Reason Codes for the IXCSYSCL Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** The request was accepted for processing. **Action:** None.
08	04	**Equate Symbol:** IXCSYSCLRSNNOTACTIVE **Meaning:** Program error. The member token specified is not for an active member in the primary address space. **Action:** The member token was returned by IXCJOIN when this member joined the XCF group. Try one or more of the following: • Verify the token passed. • Ensure that this request is running in the correct address space. • Ensure that this request is for an active member. • Verify that if IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • Verify that if member token was specified using explicit register notation, the corresponding access register was updated accordingly.

Table 16. Return and Reason Codes for the IXCSYSCL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	08	**Equate Symbol:** IXCSYSCLRSNINAPPROPRIATEPRIMARY **Meaning:** Program error. The primary address space is neither the master scheduler address space nor the primary address space of the caller of the IXCJOIN request. **Action:** This macro must be issued from the same primary address space as when the IXCJOIN was issued. Ensure that: • The parameter list has not been overlaid. • This request is running in the correct address space. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro.
08	0C	**Equate Symbol:** IXCSYSCLRSNSYSCLEANUPMEMNO **Meaning:** Program error. The member token specified in MEMTOKEN is not for a member that specified SYSCLEANUPMEM=YES on the IXCJOIN macro. **Action:** The IXCSYSCL macro should be issued only by a member that joined the XCF group with SYSCLEANUPMEM=YES specified. If this return code is unexpected, then ensure that: • The member token has not been overlaid. • The correct member token is being used. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the member token was specified using explicit register notation, the corresponding access register was updated accordingly.
08	10	**Equate Symbol:** IXCSYSCLRSNFAILEDSYSNOTVALID **Meaning:** Program error. The system token specified for FAILEDSYS is not valid. **Action:** The system token is from the group exit parameter list (IXCYGEPL). Ensure that: • The system token has not been overlaid. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the address space token was specified using explicit register notation, the corresponding access register was updated accordingly.
08	40	**Equate Symbol:** IXCSYSCLRSNPLISTRSVDNOTVALID **Meaning:** Program error. A reserved field in the parameter list is not zero. **Action:** Ensure that: • The parameter list was not inadvertently overlaid. • The parameter list was initialized before it was used. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • The correct parameter list version was specified.

Table 16. Return and Reason Codes for the IXCSYSCL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	100	**Equate Symbol:** IXCSYSCLRSNPLISTBADALET **Meaning:** Program error. The ALET that qualifies the address of the parameter list is not valid. **Action:** Ensure that: • The address of the parameter list has not been overlaid. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly.
08	104	**Equate Symbol:** IXCSYSCLRSNPLISTVERSIONNOTVALID **Meaning:** Program error. The version number specified in the IXCSYSCL parameter list is not valid. **Action:** Ensure that: • Your program did not overlay the parameter list storage. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • The correct parameter list version was specified.
08	108	**Equate Symbol:** IXCSYSCLRSNPLISTBADFUNCTION **Meaning:** Program error. The parameter list is not valid. **Action:** Ensure that: • Your program did not overlay the parameter list storage. • Your program was assembled with the correct macro library for the release of MVS your program is running on. • If IXCSYSCL was called in AR mode, the SYSSTATE ASCENV=AR macro was issued prior to this macro. • If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly.
08	10C	**Equate Symbol:** IXCSYSCLRSNPLISTBADSTG **Meaning:** Program error. An error occurred when the system tried to access the parameter list. **Action:** Take one or more of the following actions: • Ensure that the parameter list address has not been overlaid. • Ensure that the correct parameter list storage area was specified. • If your program is running in AR ASC mode ensure that: – You specified SYSSTATE ASCENV=AR before issuing the IXCSYSCL macro. – If the parameter list address was specified using explicit register notation, the corresponding access register was updated accordingly. – The parameter list is either in the primary address space or in an address/data space that is addressable through a public entry on the caller’s DU-AL. • Ensure that the parameter list storage area was not inadvertently freed by your program. • Ensure that your program was assembled with the correct macro library for the release of MVS your program is running on.

Table 16. Return and Reason Codes for the IXCSYSCL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	11C	**Equate Symbol:** IXCSYSCLRSNNOTENABLED **Meaning:** The caller is not enabled. **Action:** Correct your program so that it does not issue IXCSYSCL while it is disabled.
08	12C	**Equate Symbol:** IXCSYSCLRSNLOCKHELD **Meaning:** Program error. The caller of IXCSYSCL holds a lock. **Action:** Correct your program so that it does not issue IXCSYSCL while it is holding a lock.
10	None.	**Meaning:** System error. The system experienced an unexpected error while processing this request. **Action:** Retry the request at least once. If the problem persists, record the return and reason code, and supply them to the appropriate IBM support personnel.

Example

```
         TITLE  'XCNSGY20- Sample IXCSYSCL macro usage'
XCNSGY20 CSECT
XCNSGY20 AMODE  31
XCNSGY20 RMODE  ANY
*/* START OF SPECIFICATIONS ********************************************
*
*
*01* MODULE-NAME = XCNSGY20
*
*02*   DESCRIPTIVE-NAME = Sample IXCSYSCL macro usage
*
*   STATUS = HBB5520
*
*01* FUNCTION =
*          Sample program to illustrate use of IXCSYSCL macro by a
*        multi-system application that becomes a XCF group member and
*        needs to perform system-wide cleanup after a system leaves
*        the sysplex.
*
*02*   OPERATION =
*
*         (1) Get into Supervisor state Key 0
*
*         (2) Load group exit routine
*
*         (3) Join a XCF group and tell XCF that this group member
*             performs system-wide cleanup after a system leaves the
*             sysplex. This is done by invoking IXCJOIN macro with
*             SYSCLEANUPMEM=YES and specifying a group exit such that
*             XCF can notify us that a system has left the sysplex.
*
*         (4) Wait for our group exit to tell us that it is time to
*             do system-wide cleanup for a system that has left the
*             sysplex. This will be done by waiting on the ECB passed
*             to the group exit.
*
*         (5) Do the necessary system-wide cleanup. This example will
*             just issue a WTO saying it is doing system_wide cleanup.
*
```

```
*         (6) Tell XCF this XCF group member has completed system-wide
*             cleanup. This is accomplished by invoking the IXCSYSCL
*             macro service and telling XCF which system the member has
*             completed system-wide cleanup for.
*
*         (7) Leave the XCF group.
*
*         (8) Delete group exit routine
*
*         (9) Return to problem state key 8
*
*
***********************************************************************
*
*02*   RECOVERY-OPERATION = This program functions without recovery.
*
***********************************************************************
*
*01* NOTES =
*
*      (1) Sample install Linkedit JCL:
*
*           //LINK      EXEC PGM=IEWL,
*           //    PARM='RENT,REFR,XREF,LET,LIST,NCAL,SIZE=(750K,200K)'
*           //SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(200,20))
*           //OBJLIB   DD DSN=userid.my.obj,DISP=SHR
*           //SYSLMOD  DD DSN=lnklst.lib,DISP=OLD
*           //SYSPRINT DD SYSOUT=*
*           //SYSLIN DD *
*               INCLUDE OBJLIB(XCNSGY20)
*               ENTRY XCNSGY20
*               NAME XCNSGY20(R)
*
*      (2) Sample execution JCL to run the example out of a authorized
*          library in the linklist concatenation:
*
*           //RUNIT     EXEC PGM=XCNSGY20
*           //SYSABEND DD SYSOUT=*
*
*
*02*   DEPENDENCIES = None
*
*02*   RESTRICTIONS =
*                     None
*
*02*   REGISTER-CONVENTIONS =
*
*03*     REGISTER-USAGE = See register declarations in code
*
*02*   PATCH-LABEL = None
*
*01* MODULE-TYPE = CSECT
*
*02*   PROCESSOR = Assembler-H
*
*02*   MODULE-SIZE = See assembler External Symbol Dictionary
*
*02*   ATTRIBUTES =
*
*03*     LOCATION = User private
*
*03*     RMODE =    Any
*
*03*     TYPE  =    Reentrant
*
**********************************************************************
```

```
*
*01* ENTRY-POINT =  XCNSGY20
*
*02*   PURPOSE =  See FUNCTION section for this module.
*
*03*     OPERATION =  See OPERATION section for this module.
*
*02*   LINKAGE = BALR
*
*03*     CALLERS = Any
*
*02*   ATTRIBUTES =
*
*03*     ENTRY
*
*04*       ENABLED
*04*       STATE = Problem program
*04*       KEY = User key
*04*       AMODE = 31
*04*       LOCKS HELD = None
*04*       ASC MODE = Primary
*04*       MEMORY MODE = Primary equal to Secondary equal to Home
*04*       DISPATCH MODE = Task
*
*03*     EXECUTION
*
*04*       ENABLED
*04*       STATE = Supervisor State
*04*       KEY = Key zero
*04*       AMODE = 31
*04*       LOCKS OBTAINED = None
*04*       ASC MODE = Primary
*04*       MEMORY MODE = Primary equal to Secondary equal to Home
*02*   SERIALIZATION = None
*
*02*   INPUT = None
*
*03*     ENTRY-REGISTERS =
*
*           R0       = Irrelevant
*           R1       = Irrelevant
*           R2 - R12 = Irrelevant
*           R13      = Address of a standard save area
*           R14      = Return address
*           R15      = Entry point address
*
*02*   OUTPUT = WTOs
*
*
*02*   EXIT-NORMAL = Return to caller
*
*03*     CONDITIONS = The multisystem application has performed
*           system-wide cleanup for a system removed from the
*           sysplex.
*
*03*     EXIT-REGISTERS = N/A
*
*03*     RETURN-CODES =
*             R15 - 00  if expected return codes from IXCJOIN, IXCSYSCL
*                         and IXCLEAVE macro services
*
*                 - 08  if unexpected return codes from IXCJOIN,
*                        IXCSYSCL, and IXCLEAVE macro services
*
*02*   EXIT-ERROR =  None
*
***********************************************************************
```

```
*
*01* EXTERNAL-REFERENCES  =
*
*
*02*   ROUTINES = None
*
*02*   DATA-AREAS = None
*
*02*   CONTROL-BLOCKS =
*
*   Common  Mapping
*    Name    Macro        Usage               Full Name
*   ------  --------  -------------  ---------------------------------
*    QUAA   IXCYQUAA  Read           IXCYQUAA Macro Service ANSAREA
*                                    mappings
*
*
*
*01* MACROS-EXECUTABLE =
*                        IXCJOIN
*                        IXCLEAVE
*                        IXCSYSCL
*                        STORAGE
*                        WTO
*
*01* SERIALIZATION = None.
*
*01* MESSAGES =
*               The following WTOs may be issued to job log:
*        XCNSGY20 IXCJOIN  RETCODE=rrrrrrrr RSNCODE:ssssssss
*        XCNSGY20 NOW IN GROUP=gggggggg MEMBER=mmmmmmmmmmmmmmmm
*        XCNSGY20 DOING SYSTEM-WIDE CLEANUP FOR cccccccc
*        XCNSGY20 IXCSYSCL RETCODE=rrrrrrrr RSNCODE:ssssssss
*        XCNSGY20 IXCLEAVE RETCODE=rrrrrrrr RSNCODE:ssssssss
*
*01* ABEND-CODES =  None
*
*01* WAIT-STATE-CODES = None
*
*01* CHANGE-ACTIVITY = None
*
**** END OF SPECIFICATIONS ******************************************/
         EJECT
***********************************************************************
*                                                                     *
*   Standard entry linkage                                            *
*                                                                     *
***********************************************************************
         STM    R14,R12,12(R13)
         BALR   BASEREG1,0             Establish addressability
         USING  *,BASEREG1
         MODID  BR=YES
         LR     R3,R13                 Save callers savearea address
         STORAGE OBTAIN,ADDR=(DATAREG1),SP=0,LENGTH=DYNASIZE
         LA     DATAREG2,4095(,DATAREG1) Set Second Data Register
         USING  DYNA,DATAREG1          First Data Register
         USING  DYNA+4095,DATAREG2     Second Data Register
         ST     R3,SAVEAREA+4          Save @ of callers savearea
         ST     DATAREG1,8(,R3)        Chain our savearea to callers
         EJECT
***********************************************************************
*                                                                     *
*   Initialize variables                                              *
*                                                                     *
***********************************************************************
```

```
         MVC    EXITRC,=AL4(GOODRETC) * Initialize return code
         MVC    WTOEXEC(LENWTOS),WTOS * Copy static parmlist to dynamic
         MVC    WTOTXTD1(L'WTOTXTD1),WTOTXTS1 * Prime WTO text length

*--------------------------------------------------------------------
* Set up MEMDATA for the XCF group member
*--------------------------------------------------------------------
         XC     DATAGE(16),DATAGE        Clear area used by group exit
         LA     R3,MYECB                 Get address of ECB
         ST     R3,MYECB_PTR             Save address for group exit
         LA     R3,DATAGE                Get address of data area to
*                                        be used by group exit to pass
*                                        information back to this task
         ST     R3,DATAGE_PTR            Save address in first word of
*                                        data area to be used by group
*                                        exit to pass info back to this
*                                        task
         EJECT
**********************************************************************
* (1) Get into Key 0 , Supervisor State                              *
*                                                                    *
**********************************************************************
         MODESET KEY=ZERO,MODE=SUP
         EJECT
**********************************************************************
* (2) Load Group exit routine.                                       *
*                                                                    *
**********************************************************************
         LOAD EP=XCNSGY21
         LR   R2,R0                 Save address of group exit
         EJECT
**********************************************************************
* (3) Join a XCF group and tell XCF that this group member           *
*     performs system-wide cleanup after a system leaves the         *
*     sysplex. This is done by invoking IXCJOIN macro with           *
*     SYSCLEANUPMEM=YES and specifying a group exit such that        *
*     XCF can notify us that a system has left the sysplex.          *
*                                                                    *
**********************************************************************
         IXCJOIN GRPNAME=MYGRPNAM,   Name of XCF group to be used      +
               MEMNAME=MYMEMNAM,     Member name to be used            +
               GRPEXIT=(R2),         Group exit routine                +
               ANSLEN=QUAMLENG,      Answer area length : One QUAMEM   +
               ANSAREA=QUAMEM,       Answer area (returned QUAMEM)     +
               MEMDATA=MYMDATA,      Member data                       +
               SYSCLEANUPMEM=YES,    Member does System-wide cleanup   +
               RETCODE=SAVERET,      Return code                       +
               RSNCODE=SAVERSN,      Reason code                       +
               MF=(E,JOINPL)
*
* Issue a WTO that shows the IXCJOIN RETCODE/RSNCODE
         MVC    WTOTXTD2(L'WTOTXTD2),WTOSERVC * Set message text
         MVC    MAPSERV(8),=CL8'IXCJOIN ' * Service invoked
* Convert hex return code to printable hex
         MVC    PHEXIN(4),SAVERET    Hex return code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRETC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRETC(8),TRTBL-240 Translate to printable hex
* Convert hex reason code to printable hex
         MVC    PHEXIN(4),SAVERSN    Hex reason code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRSNC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRSNC(8),TRTBL-240 Translate to printable hex
         BAL    R14,ISSUEWTO         Tell user IXCJOIN RETCODE/RSNCODE

**********************************************************************
*     Check for successfully joining XCF group.                      *
*                                                                    *
```

```
***********************************************************************
         L      R3,SAVERET           Get return code
         C      R3,=AL4(4)           Member is now in active state
         BH     BADJOIN              Bad return code from IXCJOIN
***********************************************************************
*     Joined XCF group successfully.                                  *
*                                                                     *
***********************************************************************
         MVC   MYMEMTOK(8),QUAMTOKN  Save my XCF member token
*
* Issue a WTO that shows the XCF group and member name join was for
         MVC    WTOTXTD2(L'WTOTXTD2),WTOJOIN  * Set message text
         MVC    MAPGRPNM(8),MYGRPNAM          * Set group name
         MVC    MAPMEMNM(16),MYMEMNAM         * Set member name
         BAL    R14,ISSUEWTO         Tell user XCF group & member name
         EJECT
***********************************************************************
* (4) Wait for our group exit to tell us that it is time to           *
*     do system-wide cleanup for a system that has left the           *
*     sysplex. This will be done by waiting on the ECB passed         *
*     to the group exit.                                              *
*                                                                     *
***********************************************************************
         LA    R3,MYECB              Point to ECB to wait on
         WAIT  ECB=(R3)              Wait for group exit to tell us it
*                                    is time to do system-wide cleanup
*                                    for some other system
         SPACE 2
***********************************************************************
* (5) Do the necessary system-wide cleanup. This example will         *
*     just issue a WTO saying it is doing system_wide cleanup and     *
*     identify name of system that group exit told us we are cleaning *
*     up for.  Group exit sets the value of SYSNAME.                  *
*                                                                     *
***********************************************************************
*
* Issue a WTO that shows that member is doing system-wide cleanup
         MVC    WTOTXTD2(L'WTOTXTD2),WTOCLEAN * Set message text
         MVC    MAPSYSNM(8),SYSNAME  Copy Sysname to WTO text area
         BAL    R14,ISSUEWTO         Say doing system-wide cleanup
         EJECT
***********************************************************************
* (6) Tell XCF this XCF group member has completed system-wide        *
*     cleanup. This is accomplished by invoking the IXCSYSCL          *
*     macro service and telling XCF which system the member has       *
*     completed system-wide cleanup for.                              *
*                                                                     *
***********************************************************************
         IXCSYSCL MEMTOKEN=MYMEMTOK, XCF group member token            +
               FAILEDSYS=MYSYSTOK,   Failed system's system token      +
               RETCODE=SAVERET,      Return code from IXCSYSCL         +
               RSNCODE=SAVERSN,      Reason code from IXCSYSCL         +
               MF=(E,SYSCLPL)
*
* Issue a WTO that shows the IXCSYSCL RETCODE/RSNCODE
         MVC    WTOTXTD2(L'WTOTXTD2),WTOSERVC * Set message text
         MVC    MAPSERV(8),=CL8'IXCSYSCL' * Service invoked
* Convert hex return code to printable hex
         MVC    PHEXIN(4),SAVERET    Hex return code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRETC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRETC(8),TRTBL-240 Translate to printable hex
* Convert hex reason code to printable hex
         MVC    PHEXIN(4),SAVERSN    Hex reason code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRSNC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRSNC(8),TRTBL-240 Translate to printable hex
         BAL    R14,ISSUEWTO         Tell user IXCSYSCL RETCODE/RSNCODE
         EJECT
***********************************************************************
*     Check for successfully telling XCF that this member has         *
*     completed system-wide cleanup for the failed system.            *
***********************************************************************
         L      R3,SAVERET           Get return code
         C      R3,=AL4(0)           Request was accepted by XCF
         BE     LEAVEGRP             Good return code from IXCSYSCL
***********************************************************************
*     XCF did not accept the IXCSYSCL macro response.                 *
***********************************************************************
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         EJECT
***********************************************************************
* (7) Leave the XCF group.                                            *
*                                                                     *
***********************************************************************
LEAVEGRP EQU *                       Label to leave XCF group
         IXCLEAVE MEMTOKEN=MYMEMTOK, XCF member token                   +
               RETCODE=SAVERET,      Return code from IXCLEAVE         +
               RSNCODE=SAVERSN       Reason code from IXCLEAVE         +
               MF=(E,LEAVEPL)
*
* Issue a WTO that shows the IXCLEAVE RETCODE/RSNCODE
         MVC    WTOTXTD2(L'WTOTXTD2),WTOSERVC * Set message text
         MVC    MAPSERV(8),=CL8'IXCLEAVE' * Service invoked
* Convert hex return code to printable hex
         MVC    PHEXIN(4),SAVERET    Hex return code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRETC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRETC(8),TRTBL-240 Translate to printable hex
* Convert hex reason code to printable hex
         MVC    PHEXIN(4),SAVERSN    Hex reason code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRSNC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRSNC(8),TRTBL-240 Translate to printable hex
         BAL    R14,ISSUEWTO         Tell user IXCLEAVE RETCODE/RSNCODE
         B      DELGEXIT             Delete the group exit
BADJOIN  EQU *                       IXCJOIN got a bad return code
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         EJECT
***********************************************************************
* (8) Delete Group exit routine                                       *
*                                                                     *
***********************************************************************
DELGEXIT EQU    *
         DELETE EP=XCNSGY21
         EJECT
***********************************************************************
* (9) Return to Key 8, Problem State                                  *
*                                                                     *
***********************************************************************
COMPLETE EQU    *
         MODESET KEY=NZERO,MODE=PROB
         EJECT
***********************************************************************
*     Exit linkage                                                    *
*                                                                     *
***********************************************************************
         L      R2,SAVEAREA+4        Save caller's save area address
         L      R3,EXITRC            Save testcase return code
         STORAGE RELEASE,ADDR=((DATAREG1)),LENGTH=DYNASIZE
         LR     R13,R2               Restore caller's save area address
         L      R14,12(R13)          Restore Return address
         LR     R15,R3               Set testcase return code
         LM     R0,R12,20(R13)       Restore Registers R0-R12
         BR     R14                  Return to caller
***********************************************************************
*                                                                     *
*   Subroutine: ISSUEWTO                                              *
*                                                                     *
*   Function  : This routine is called to issue a WTO.                *
*                                                                     *
*   Input     : WTOTXTD1 contains text for WTO message to be issued   *
*                                                                     *
*                                                                     *
***********************************************************************
ISSUEWTO EQU *
         STM    R14,R12,SAVE1          Save callers regs
         LA     R5,WTOTXTD1            Address WTO parmlist
         WTO TEXT=(R5),ROUTCDE=(11),MF=(E,WTOEXEC) * Issue WTO
         LM     R14,R12,SAVE1          Restore callers regs
         BR     R14                    Return to caller
         EJECT
***********************************************************************
*                                                                     *
*   Register declares                                                 *
*                                                                     *
***********************************************************************
R0       EQU    0
R1       EQU    1
R2       EQU    2
R3       EQU    3
R4       EQU    4
R5       EQU    5
R6       EQU    6
R7       EQU    7
R8       EQU    8
R9       EQU    9
R10      EQU    10                     Reserved for future expansion
*                                      of the code or the dynamic area
DATAREG2 EQU    11                     Second data register
BASEREG1 EQU    12                     Code register
R12      EQU    12
DATAREG1 EQU    13                     First data register
R13      EQU    13
R14      EQU    14
R15      EQU    15
         EJECT
***********************************************************************
*                                                                     *
*   Static data                                                       *
*                                                                     *
***********************************************************************
         DS     0F
TRTBL    DC     CL16'0123456789ABCDEF' * Translate table
ZERO     DC     F'0'                   * Constant zero for comparisons
         SPACE  1
***********************************************************************
*                                                                     *
*   Static WTO data                                                   *
*                                                                     *
***********************************************************************
WTOS     WTO TEXT=,ROUTCDE=(11),MF=L * Static form of WTO
LENWTOS  EQU *-WTOS                   * Length of WTO parmlist
WTOTXTS1 DC     AL2(L'WTOTXTD2)       * WTO text length
WTOSERVC DC CL65'XCNSGY20 mmmmmmmm RETCODE=rrrrrrrr RSNCODE=ssssssss'
MAPSERV  EQU    WTOTXTD2+9,8,C'C'     * Map service WTO insert
MAPRETC  EQU    WTOTXTD2+26,8,C'C'    * Map service RETCODE insert
MAPRSNC  EQU    WTOTXTD2+43,8,C'C'    * Map service RETCODE insert
WTOCLEAN DC CL65'XCNSGY20 DOING SYSTEM-WIDE CLEANUP FOR cccccccc     '
MAPSYSNM EQU    WTOTXTD2+39,8,C'C'    * Map system name insert
```

```
WTOJOIN DC CL65'XCNSGY20 NOW IN GROUP=gggggggg MEMBER=mmmmmmmmmmmmmmmm'
MAPGRPNM EQU    WTOTXTD2+22,8,C'C'    * Map group name insert
MAPMEMNM EQU    WTOTXTD2+38,8,C'C'    * Map member name insert
         EJECT
***********************************************************************
*                                                                     *
*   Constants used to identify the application to XCF.                *
*                                                                     *
***********************************************************************
MYGRPNAM DC     CL8'XCNSGY20'          Application group name to join
*                                      XCF group with
MYMEMNAM DC     CL16'M1              ' Application member name to join
*                                      XCF group with
*                                       WTOTXTD2 fields)
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*   Dynamic data                                                      *
*                                                                     *
***********************************************************************
DYNA     DSECT
SAVEAREA DS     18F                    Standard savearea (first field)
         SPACE  1
***********************************************************************
*   Member data passed to group exit via MEMDATA keyword              *
***********************************************************************
            DS  0D                     Ensure boundary alignment
MYMDATA     DS  CL8                    Member data provided to group
*                                      when member needs to do
*                                      system-wide cleanup
DATAGE_PTR EQU MYMDATA+0,4,C'A'        Address of data area to be used
*                                      by group exit to pass info back
*                                      to this task
         SPACE  1
***********************************************************************
*   Data area to be used by group exit to pass info back to this task *
***********************************************************************
            DS  0D                     Ensure boundary alignment
DATAGE      DS  CL16                   Data area to be used by
*                                      group exit to pass info back to
*                                      this task
SYSNAME      EQU DATAGE+0,8,C'C'       System name being cleanup up for
MYSYSTOK     EQU DATAGE+8,4,C'F'       System token to cleanup for
MYECB_PTR    EQU DATAGE+12,4,C'A'      ECB that group exit should post
         SPACE  1
            DS  0D                     Ensure boundary alignment
MYGE_ADDR   DS  A                      Address of group exit
MYMEMTOK    DS  XL8                    Member token of XCF group member
PHEXIN   DS   CL5                      Work area for printable hex conv
PHEXOUT  DS   CL10                     Work area for printable hex conv
MYECB    DS     F                      ECB for group exit to post
SAVERET  DS     F                      Save macro service return code
SAVERSN  DS     F                      Save macro service reason code
SAVE1    DS     15F                    First level subroutine savearea
SAVE2    DS     15F                    Second level subroutine savearea
EXITRC   DS     F                      Module Return code
BADRETC  EQU    8                      Bad return code from testcase
GOODRETC EQU    0                      Good return code from testcase
         EJECT
***********************************************************************
*                                                                     *
*   Dynamic WTO storage                                               *
*                                                                     *
***********************************************************************
WTOEXEC  WTO TEXT=,ROUTCDE=(11),MF=L * List form of WTO parmlist
```

```
WTOTXTD1 DC     AL2(L'WTOTXTD1)       * WTO text length
WTOTXTD2 DC     CL65' '               * WTO text
         SPACE 2
***********************************************************************
*   Dynamic storage for parmlists                                     *
***********************************************************************
         SPACE 2
         IXCJOIN MF=(L,JOINPL)
         IXCSYSCL MF=(L,SYSCLPL)
         IXCLEAVE MF=(L,LEAVEPL)
***********************************************************************
*   Dynamic storage for a answer area to hold a QUAMEM record         *
*     recorded by IXCJOIN.                                            *
***********************************************************************
         IXCYQUAA DSECT=NO,HEADER=NO,MEMBER=YES,SYSTEM=NO,             +
               GROUP=NO,CF=NO,CFSC=NO,CFSTR=NO,STR=NO,                 +
               STRPL=NO,STRXL=NO,STRCF=NO,STRUSER=NO,ARMS=NO
         SPACE 2
***********************************************************************
*                                                                     *
*   End of dynamic storage                                            *
*                                                                     *
***********************************************************************
DYNASIZE EQU    *-DYNA                 Total size of dynamic storage
         EJECT
         END
```

IXCTERM — Terminate a Member of an XCF Group

Description

The IXCTERM macro terminates an active member (called the target member) of a cross-system coupling facility (XCF) group. The system abnormally ends the associated task (either the task that issued the IXCJOIN causing the target member to become active, or the jobstep task that was current at the time the IXCJOIN was issued. Which type of task the member is associated with is specified the MEMASSOC parameter on the IXCJOIN.)

Note: IXCTERM does not terminate a member that is address space associated. If the target member is associated with an address space, the task that issued the IXCTERM will receive return code X'8', reason code X'120'.

The abnormal end might cause other members who were associated with the same task to terminate. XCF does not allow the recovery routines for that task to retry. Use the IXCTERM macro to remove from the XCF group another active member of the same group. The actual termination of the target member occurs asynchronously.

After the target member terminates, XCF notifies the remaining members in the group that have defined a group user-routine that the member is no longer associated with the group. The target member's recovery routine determines the new state of the target member: quiesced, failed, or not-defined.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0-7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space of the caller of the IXCJOIN that placed the calling member in the active state.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXCTERM. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

- The caller and the target member must be active members of the same XCF group.
- The caller can have no enabled, unlocked task (EUT) FRRs established.
- The target member cannot be address space associated.

Input Register Information

Before issuing the IXCTERM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	If GPR 15 contains a zero or X'10', GPR 0 is used as a work register by the system; otherwise, GPR 0 contains a reason code.
1	Used as a work register by the system.
2-13	Unchanged.
14	Used as a work register by the system.
15	Return code.

When control returns to the caller, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

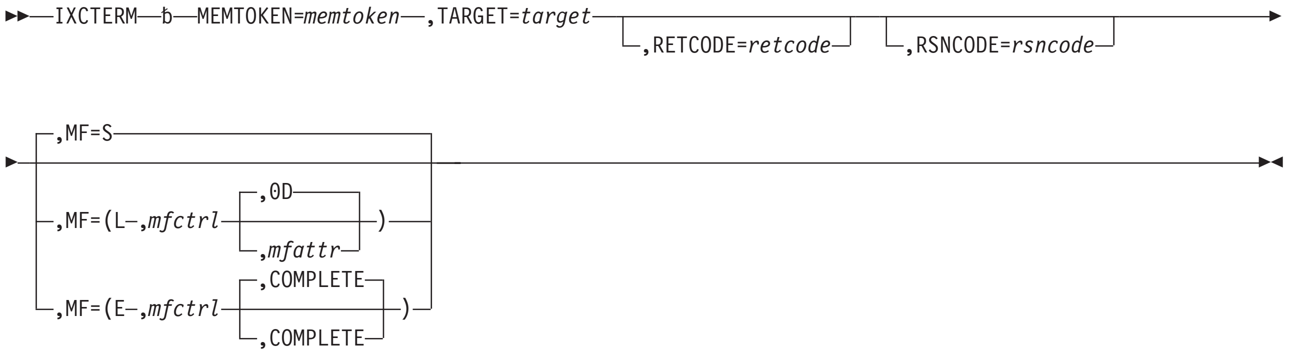
Performance Implications

None.

Syntax Diagram

The syntax of the IXCTERM macro is as follows:

IXCTERM diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

MEMTOKEN=*memtoken*

Use this input parameter to specify the 64-bit token of the member issuing IXCTERM. IXCJOIN provided this token when it activated the member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token for the member issuing the IXCTERM.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**

,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request is completed.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request is completed.

,TARGET=*target*

Use this input parameter to specify the 64-bit token of the active member that XCF is to terminate. IXCJOIN provided this token when it activated the member.

Note: The target member must be a member of the same XCF group as the issuer of the IXCTERM. The target member cannot be an address space associated member.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the member token of the member XCF is to terminate.

ABEND Codes

None.

Return and Reason Codes

When the IXCTERM macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXCYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXCRETCODEOK
4	IXCRETCODEWARNING
8	IXCRETCODEPARMERROR
C	IXCRETCODEENVERROR
10	IXCRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 17. Return and Reason Codes for the IXCTERM Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
00	None.	**Meaning:** IXCTERM completed successfully; XCF will terminate the member. **Action:** None.
08	04	**Equate Symbol:** IXCTERMRSNNOTACTIVE **Meaning:** Program error. The calling member token does not identify an active member. **Action:** Take one or more of the following actions: • Ensure that the proper MEMTOKEN address and ALET (if appropriate) were used. • Any additional action depends on your application. Because the member (MEMTOKEN) is no longer active, your program might want to stop any further activity associated with the member.
08	08	**Equate Symbol:** IXCTERMRSNINAPPROPRIATEPRIMARY **Meaning:** Program error. The primary address space is not the same as the primary address space of the caller of the IXCJOIN that placed the calling member in the active state. **Action:** Take one or more of the following actions: • Ensure that the correct MEMTOKEN address and ALET (if appropriate) were used. • Change your program to issue IXCTERM from the address space in which the calling member joined the group and retry the request.

Table 17. Return and Reason Codes for the IXCTERM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	0C	**Equate Symbol:** IXCTERMRSNTARGETNOTACTIVE **Meaning:** Program error. The target member is not an active member. **Action:** If the member is expected to be in an active state, ensure that the TARGET MEMTOKEN is correct, and retry the request. If the target member is required to be in a not-defined state, you can use the IXCQUERY service to determine if the member is in a created, quiesced, or failed state. You can use the IXCDELET service to place the member in a not-defined state.
08	10	**Equate Symbol:** IXCTERMRSNTARGETNOTDEFINED **Meaning:** Program error. The target member is not defined to XCF. **Action:** If the TARGET member is expected to be in an active, created, quiesced, or failed state, ensure that the correct TARGET MEMTOKEN was specified. Otherwise, no action is required because the target member has terminated.
08	14	**Equate Symbol:** IXCTERMRSNTARGETDIFFERENTGROUP **Meaning:** Program error. The target member and the issuing member are in different XCF groups. **Action:** Take one or more of the following actions: • Ensure that the correct TARGET and MEMTOKEN member tokens were specified. Check the addresses and ALETs (if appropriate). • You might have to change your program to ensure that it issues IXCTERM only with TARGET and caller (MEMTOKEN) members that are in the same group.
08	18	**Equate Symbol:** IXCTERMRSNTARGETNOTVALID **Meaning:** Program error. The target member token is not valid. **Action:** Take one or more of the following actions: • Ensure that the correct TARGET member token address and ALET (if appropriate) were used. • If your program is running in AR mode, ensure that the SYSSTATE ASCENV=AR macro was issued before the IXCTERM.
08	1C	**Equate Symbol:** IXCTERMRSNMEMTOKENNOTVALID **Meaning:** Program error. The token of the issuing member is not valid. **Action:** Correct the member token and retry the request.
08	40	**Equate Symbol:** IXCTERMRSNPLISTRSVDNOTVALID **Meaning:** Program error or environmental error. A reserved field in the control parameter list is not zero. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.

Table 17. Return and Reason Codes for the IXCTERM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
08	100	**Equate Symbol:** IXCTERMRSNPLISTBADALET **Meaning:** Program error. Your program is running in AR mode, and the ALET that qualifies the address of the control parameter list is neither zero nor associated with a valid entry on the caller's DU-AL. **Action:** Ensure that: • You specified SYSSTATE ASCENV=AR before issuing the IXCJOIN macro. • The ALET for the parameter list is on the DU-AL or is zero (primary address space ALET).
08	104	**Equate Symbol:** IXCTERMRSNPLISTVERSIONNOTVALID **Meaning:** Program error. The version number in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage.
08	108	**Equate Symbol:** IXCTERMRSNPLISTBADFUNCTION **Meaning:** Program error. The function code in the control parameter list is not valid. **Action:** Check to see if your program inadvertently overlaid the parameter list storage, and that it was assembled with the correct macro library for the release of MVS your program is running on.
08	10C	**Equate Symbol:** IXCTERMRSNPLISTBADSTG **Meaning:** Program error or environmental error. XCF could not access the control parameter list. **Action:** Take one or more of the following actions: • Ensure that the correct parameter list storage area was specified. • If your program is running in AR mode: – Ensure that you specified SYSSTATE ASCENV=AR before issuing the IXCTERM macro. – Ensure that the parameter list ALET is correct. • Ensure that the parameter list storage area was not inadvertently freed by your program.
08	118	**Equate Symbol:** IXCTERMRSNNOTTASKMODE **Meaning:** Program error. The caller is not in task mode. **Action:** Correct your program so that it issues IXCTERM only while in task mode.
08	11C	**Equate Symbol:** IXCTERMRSNNOTENABLED **Meaning:** Program error. The caller is not enabled. **Action:** Correct your program so that it issues IXCTERM only while enabled.

Table 17. Return and Reason Codes for the IXCTERM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
08	120	**Equate Symbol:** IXCTERMRSNTARGETNOTMEMASSOCTASK **Meaning:** Program error. The target member cannot be terminated because it is associated with an address space instead of a task. **Action:** Change your program to associate the member with a task, or cancel the member's associated address space.
10	None.	**Meaning:** System error. XCF processing failed. **Action:** Retry the request at least once. If the problem persists, record the return code, and supply it to the appropriate IBM support personnel.

Example

Operation: Terminate a member from a group. The token of the caller is TOKEN1; the token of the target member is TOKEN2. XCF is to store the return code and reason code into the variables RETURN and REASON. The code is as follows:

```
         IXCTERM  MEMTOKEN=TOKEN1,TARGET=TOKEN2,                      X
              RETCODE=RETURN,RSNCODE=REASON,MF=S

TOKEN1    DS   CL8                     TOKEN OF MEMBER ISSUING IXCTERM
TOKEN2    DS   CL8                     TOKEN OF MEMBER TO BE          X
                                       TERMINATED
RETURN    DS   1F                      RETURN CODE
REASON    DS   1F                      REASON CODE
```

You can access the member tokens from the QUAMTOKN field in the area returned by IXCJOIN and IXCQUERY, and mapped by the IXCYQUAA mapping macro.

IXLALTER — Alter a Coupling Facility Structure

Description

The IXLALTER service allows an authorized program, not necessarily a connected user of a structure, to start or stop the structure alter process. Structure alter is a non-disruptive process to expand or contract the size of a structure, to reapportion the structure's entry-to-element ratio, and to change the percentage of structure storage that is set aside for event monitor controls (EMCs).

The structure to be altered must be allocated in a coupling facility that supports structure alter, that is, a coupling facility with CFLEVEL=1 or higher. For keyed list structures containing EMCs in a coupling facility with CFLEVEL=3, you can alter the size or the entry-to-element ratio of the structure, but the EMC storage percentage will not change. For keyed list structures containing EMCs in a coupling facility with CFLEVEL=4 or higher, you can also alter the EMC storage percentage.

A structure alter can be done only if all connectors to the structure allow structure alter (indicated by specifying ALLOWALTER=YES on IXLCONN). At connect time, connectors can also specify whether changing the entry-to-element ratio and EMC storage percentage is permitted (RATIO=YES or NO). If a change is permitted, the connector can also indicate the minimum threshold levels for entries, elements, and EMC storage (MINENTRY, MINELEMENT, MINEMC). These minimum threshold levels indicate the percent of entries, elements, and EMC storage that must be available when the structure alter completes.

- MINENTRY and MINELEMENT

  For list structures, this is the percent of "in-use" entries and/or elements; for cache structures, this is the percent of "in-use and changed" entries and/or elements.
- MINEMC

  For keyed list structures in a coupling facility with CFLEVEL=4 or higher, this is the percent of "currently-in-use" EMCs.

A structure alter request will not be accepted for:

- A structure that is already in the rebuild or alter process.
- A structure that is in a user-managed or system-managed duplexing rebuild but not in the Duplex Established phase.

If a structure is in user-managed duplexing rebuild processing, the alter is automatically applied to both structure instances serially. The primary (old) structure is altered first, with corresponding Alter Begin and Alter End events. After the Alter End is delivered for the primary structure, alter begins for the secondary (new) structure, also with its corresponding Alter Begin and Alter End events. The alter events indicate which of the structures is currently being altered. The system will not allow you to start a subsequent alter request for the structure until both instances of the structure have completed the current alter process.

If a structure is in system-managed duplexing rebuild processing, the alter is applied to both structure instances serially, with the primary (old) structure being altered first followed by the alter of the secondary (new) structure. However, only one set of Alter Begin and Alter End events are delivered. The Alter Begin event is delivered at the start of alter processing for the primary structure; the Alter End event is delivered at the end of alter processing for both instances of the structure.

Alter processing ends when:

- The target size, ratio, or EMC storage percentage is satisfied, either completely or to the extent possible based on available coupling facility resources. (When no more progress can be made toward attaining requested targets, IXLALTER processing ends.)
- A stop alter request is received (either a SETXCF command or an IXLALTER request to stop the alter). This does not apply to structures in the Duplex Established phase of a system-managed duplexing rebuild. While an alter of a structure in the Duplex Established phase of system-managed duplexing

rebuild is in progress, the system will reject an attempt to stop the alter with either message IXC531I (for SETXCF) or return code IXLRSNCODEDUPALTER (for IXLALTER).

- A structure or coupling facility failure occurs. If the SP 5.2 or higher systems in the sysplex all fail or lose connectivity to the coupling facility in which the structure resides, alter processing ends when the first SP 5.2 or higher system regains connectivity to the coupling facility.
- A structure rebuild request is received, except when the alter processing is for a structure in the Duplex Established phase of user-managed or system-managed duplexing rebuild. In that case, the system does not accept the structure rebuild request, and the alter processing continues.
- A request to stop duplexing rebuild is received while the structure is being altered in the Duplex Established phase. Alter processing depends on which structure instance is to be kept and which is being deallocated, and on the amount of alter processing that has been performed or remains to be performed. If the alter of the duplexed structure has already completely processed the structure that is to be kept in simplex mode, then the alter processing is finished and the system issues the Alter End event. If there is still alter processing either in progress against, or not yet performed against, the structure that is to be kept, then the alter processing continues against the simplex structure after duplexing is switched or stopped. When that alter processing against the simplex structure completes, the system issues the Alter End event.

When the alter processing is complete, all active connectors are notified of the completion through their event exit.

The security administrator may be using RACF or another security product to restrict the use of installation coupling facility structures. If so, ensure that you are authorized to issue the IXLALTER macro for the structure.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Input Register Information

Before issuing the IXLALTER macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLALTER macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if GPR 15 contains a non-zero return code
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLALTER macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLALTER. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Include the IXLYCON macro in your program. This macro provides a list of equate symbols for users of XES services and exits.

Performance Implications

IXLALTER processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLALTER Version Support

The IXLALTER macro supports versions in the range of 0 - 1.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLALTER macro.
- The following keywords are supported by all versions starting with version 1 and higher of the IXLALTER macro.

EMCSTG	EMCSTGPCT

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Note that no special version of IXLALTER is required when altering a structure in the Duplex Established phase of structure duplexing.

Syntax Diagram

The syntax of IXLALTER is written as follows:

main diagram

```
►►──IXLALTER──b──STRNAME=strname──,REQUEST=─┬─START─┤ parameters-1 ├─┬──┬──────────────────┬──────────────►
                                            └─STOP──────────────────┘  └─,RETCODE=retcode─┘
```

IXLALTER Macro

```
                            ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S──────────────────────────────────┐
►──┬──────────────────┬─────┼───────────────────────────┼───┼────────────────────────────────────────┼──────────►◄
   └─,RSNCODE=rsncode─┘     ├─,PLISTVER=MAX─────────────┤   │                   ┌─,0D──────┐         │
                            └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl────┼──────────┼─)───────┤
                                                            │                   └─,mfattr──┘         │
                                                            │                   ┌─,COMPLETE─┐        │
                                                            └─,MF=(E─,mfctrl────┼───────────┼─)──────┘
                                                                                └─,COMPLETE─┘
```

parameters-1

```
        ┌──────────────────────────────────────────────────────────┐
        ▼                              (1)                         │
►►──────┬─,SIZE─,STRSIZE=strsize──────────────────────────────────┬┴──────────────────────────►◄
        ├─,RATIO─,ENTRYRATIO=entryratio─,ELEMENTRATIO=elementratio─┤
        └─,EMCSTG─,EMCSTGPCT=emcstgpct────────────────────────────┘
```

Notes:

1 Choose each keyword (SIZE/RATIO/EMCSTG) no more than once.

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ELEMENTRATIO=*elementratio*

Use this input parameter to specify the data element part of the entry-to-element ratio.

- A cache structure initially allocated with data elements can be altered to contain no data elements and then changed again to contain data elements.
- A list structure initially allocated with data elements cannot be altered to contain no data elements.
- Cache or list structures initially allocated without data elements cannot be altered to contain data elements.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword field that contains the element part of the ratio.

,EMCSTG

Use this input parameter to indicate that the percentage of available storage to be set aside for event monitor controls (EMCs) is to be changed.

When EMCSTG is not specified, it is possible to have the EMC storage percent change as a result of a SIZE request unless at least one connector specified RATIO=NO at connect time.

This parameter is valid only for keyed list structures allocated in a coupling facility with CFLEVEL=4 or higher.

,EMCSTGPCT=*emcstgpct*

Use this input parameter to specify the percentage of available list structure storage that is to be set aside for event monitor controls (EMCs).

Specify the percentage value in hundredths of a percent. For example, to request a value of 20%, code EMCSTGPCT=2000.

The value of EMCSTGPCT must be in the range of from 0 to 10000.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword field that contains a value to express the percentage of available structure storage that is to be set aside for EMCs.

,ENTRYRATIO=*entryratio*

Use this input parameter to specify the entry part of the entry-to-element ratio.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword input field that contains the entry part of the ratio.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLALTER Version Support" on page 235 for a description of the options available with PLISTVER.

,RATIO
Use this input parameter to indicate that the entry-to-element ratio of the named structure is to be reapportioned.

When RATIO is not specified, it is possible to have the entry-to-element ratio change as a result of a SIZE request unless at least one connector specified RATIO=NO at connect time.

,REQUEST=START
,REQUEST=STOP
Use this input parameter to identify the type of structure alter request.

START
Start structure alter processing for the specified structure.

STOP
Stop alter processing for the specified structure.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,SIZE
Use this input parameter to indicate that the structure is to be expanded or contracted.

Note: When a structure is allocated, its maximum structure size (MSS) is set equal to the structure size specified in the CFRM active policy. The actual structure size may be less than or equal to the MSS value. It may be less due to one or more of the following:

- INITSIZE value in the CFRM active policy
- STRSIZE specified on the IXLCONN macro
- Storage constraints in the coupling facility
- A previous structure alter.

The actual structure size may change with the MSS value being the upper bound. The MSS value remains constant as long as this instance of the structure remains allocated.

The minimum structure size (MINSS) is the minimum control space required to allocate the structure with the attributes specified. The MINSS value is established by the coupling facility based on the attributes specified when the structure was allocated. The MINSS value can change when the structure is reapportioned with an entry-to-element ratio that differs substantially from the previous ratio. MINSIZE specified in the CFRM active policy will serve as a minimum bound for the structure size on all structure allocation requests (including alter, connect, and rebuild connect) except for new structure allocation during System-managed rebuild. New structure allocation during System-managed rebuild may cause the structure to be allocated with a size smaller than MINSIZE in the CFRM active policy. The actual structure size may expand to the MSS value or contract to the MINSS value or the MINSIZE value specified in the CFRM policy. When contracting, the larger of the two values will be used.

When SIZE is not specified the structure size will not change as a result of the IXLALTER.

,STRNAME=*strname*
Use this input parameter to specify the name of the target structure for alter start or stop processing. The name must be defined in the CFRM (coupling facility resource management) active policy.

The structure name must be 16 characters long, padded on the right with blanks if necessary. It may contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_). It must begin with an uppercase alphabetic character. IBM names begin with SYS, an IBM component prefix, or letters A-I.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the structure name.

,STRSIZE=*strsize*

Use this input parameter to specify the desired size of the structure in units of 4K (4096 bytes) blocks.

See *z/OS MVS Programming: Sysplex Services Guide* for information about minimum and maximum structure size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of 4K (4096 bytes) blocks to make up the structure.

Return and Reason Codes

When the IXLALTER macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, if applicable, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 18. Return and Reason Codes for the IXLALTER Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** IXLALTER request accepted. **Action:** None.
4	xxxx0427	**Equate Symbol:** IXLRSNCODEALREADYALTERING **Meaning:** The request was rejected because alter is already in progress for the structure. A new alter request will not be accepted until the current alter completes or is stopped. **Action:** Either wait for the structure alter to complete or issue IXLALTER to stop the structure alter.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The IXLALTER parameter list is not accessible. **Action:** Verify that: • The parameter list address is uncorrupted • The parameter list is addressable in the caller's primary address space. • If you are calling IXLALTER in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLALTER in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLALTER.

Table 18. Return and Reason Codes for the IXLALTER Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** Program error. The IXLALTER parameter list ALET is not valid. **Action:** Verify that: • If you issued IXLALTER in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLALTER in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing the IXLALTER. • If you are not running in AR-mode, and the parameter list does not need to be ALET-qualified, the corresponding access register should contain zeros.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** Program error. The IXLALTER parameter list version number is not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** Program error. The requestor is in SRB mode. **Action:** Change into task mode before issuing IXLALTER.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. The requestor is not in an enabled state. **Action:** The requestor must be enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Program error. The requestor's primary address space is not the same as the home address space. **Action:** The primary address space must equal the home address space to issue the IXLALTER.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. EMCSTGPCT cannot be altered on a cache or lock structure. **Action:** EMCSTGPCT is meaningful only for a list structure.
8	xxxx084C	**Equate Symbol:** IXLRSNCODENOSAFAUTH **Meaning:** Program error. The requestor does not have proper SAF authorization. **Action:** Have the installation set up the proper SAF authorization for access to the structure.

Table 18. Return and Reason Codes for the IXLALTER Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0883	**Equate Symbol:** IXLRSNCODEBADEMCSTGPCT **Meaning:** Program error. The value specified for EMCSTGPCT is not valid. The value must be between 0 and 10000. **Action:** Correct the value of EMCSTGPCT.
C	xxxx0C05	**Equate Symbol:** IXLRSNCODESTRNOTINPOLICY **Meaning:** Environmental error. The requested structure is not in the CFRM active policy. **Action:** Specify a structure that is currently defined in the CFRM active policy or switch to a new CFRM policy that contains the structure for which this IXLALTER was invoked.
C	xxxx0C0A	**Equate Symbol:** IXLRSNCODESTRNOTALLOCATED **Meaning:** Environmental error. The requested structure is not allocated. **Action:** Ensure that you have specified the correct structure name on the IXLALTER request. If so, allocate the structure by issuing the appropriate IXLCONN request.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. Structure failure has occurred. **Action:** None.
C	xxxx0C29	**Equate Symbol:** IXLRSNCODEXESNOTACTIVE **Meaning:** Environmental error. The CFRM function is not active or not available. **Action:** Bring the CFRM couple data set into use in the sysplex by issuing the SETXCF COUPLE,TYPE=CFRM,PCOUPLE= command.
C	xxxx0C36	**Equate Symbol:** IXLRSNCODESTOPINPROGRESS **Meaning:** Environmental error. Request rejected because structure rebuild stop was in progress for the structure. **Action:** Wait to initiate structure alter until the structure rebuild stop process completes.
C	xxxx0C51	**Equate Symbol:** IXLRSNCODEREBUILDINPROGRESS **Meaning:** Environmental error. Request rejected because structure rebuild was in progress for the structure. **Action:** Wait to initiate structure alter until the structure rebuild process completes or stop the rebuild process.
C	xxxx0C60	**Equate Symbol:** IXLRSNCODENOALTERCF **Meaning:** Environmental error. Request rejected because the structure is allocated in a coupling facility that does not support alter. CFLEVEL is equal to zero. **Action:** Reallocate the structure in a coupling facility with CFLEVEL=1 or higher.

Table 18. Return and Reason Codes for the IXLALTER Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C61	**Equate Symbol:** IXLRSNCODEALLOWALTER **Meaning:** Environmental error. Request rejected because at least one active, disconnecting, failing, or failed-persistent connection specified ALLOWALTER=NO on IXLCONN. **Action:** Use IXCQUERY to determine the set of connections that do not support structure alter for the structure. When those connections are no longer connected to the structure, retry the IXLALTER request. If a connector to the structure is failed-persistent, consider the use of IXLFORCE to delete the connection. Be aware that such an action might result in the loss of persistent information.
C	xxxx0C62	**Equate Symbol:** IXLRSNCODEALTERRATIOCHG **Meaning:** Environmental error. Request rejected because at least one active, failing, or failed-persistent connection specified RATIO=NO on IXLCONN. **Action:** Use IXCQUERY to determine the set of connections that specified RATIO=NO for the structure. When those connections are no longer connected to the structure, retry the IXLALTER request. If a connector to the structure is failed-persistent, consider the use of IXLFORCE to delete the connection. Be aware that such an action might result in the loss of persistent information.
C	xxxx0C63	**Equate Symbol:** IXLRSNCODEALTERNOTINPROG **Meaning:** Environmental error. Request rejected because structure alter stop was requested and structure alter is not in progress for the structure. **Action:** None.
C	xxxx0C66	**Equate Symbol:** IXLRSNCODEALTERSTOPINPROG **Meaning:** Environmental error. Request rejected because alter stop was requested and an alter stop is already in progress. **Action:** None.
C	xxxx0C76	**Equate Symbol:** IXLRSNCODEDUPALTER **Meaning:** Environmental error. An IXLALTER request was rejected because alter stop was requested and a stop of an alter for a structure in a system-managed duplexing rebuild is not allowed. **Action:** None.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. There are no coupling facility services available. The hardware support necessary to provide coupling facility services might not be present. **Action:** Re-IPL the system, or follow your particular failure management protocol.

Table 18. Return and Reason Codes for the IXLALTER Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	—	**Meaning:** XES processing has failed. **Action:** Save the reason code information and contact the IBM support center.

IXLCACHE — Cache Services

Description

A cache structure is a coupling facility structure that enables high-performance sharing of cached data by applications in a sysplex. The IXLCACHE macro provides the means for applications in a sysplex to access and manipulate the data in a coupling facility cache structure.

The IXLCACHE macro has many REQUEST types. These requests enable the cache structure user to manage, manipulate, and share data with other connected users of the cache structure. Each request type is detailed in the following sections. Be sure to read the IXLCACHE guidance information in *z/OS MVS Programming: Sysplex Services Guide*. The information presented here assumes you have done so. The following table tells you where to find the reference information for each request type:

Table 19. Request Types for IXLCACHE

Request Type	Description	Page
CASTOUT_DATA	Read a data item from an entry in the cache structure to the storage areas specified by BUFFER or BUFLIST and/or ADJAREA. From these areas, the data can be written to permanent storage such as DASD.	251
CASTOUT_DATALIST	Read data items from 1 to 8 entries in the cache structure to the storage areas specified by BUFFER,BUFLIST, DEIBAREA,and/or ADJAREA. From these areas, the data can be written to permanent storage such as DASD.	271
CROSS_INVAL	Deregisters interest and invalidates the copies of a data item in the local cache buffers of users with registered interest in the data item other than the requesting user.	291
CROSS_INVALLIST	Deregisters interest and invalidates the copies of from 1 to 4096 data items in the local cache buffers of users with registered interest in the data item other than the requesting user.	303
DELETE_NAME	Delete one or more data items identified by NAME and NAMEMASK from the coupling facility cache structure (this frees directory and data entry resources and invalidates any registered users).	321
DELETE_NAMELIST	Delete a set of data entries from the coupling facility cache structure. Each entry to be deleted is represented by a name block contained in the storage area specified by BUFFER or BUFLIST.	335
PROCESS_REFLIST	Process one or more cached data items to indicate that they are recently referenced.	353
READ_COCLASS	Read directory entry names and directory entry user data for the data items specified by NAME and NAMEMASK, and the cast-out class specified by COCLASS and store in BUFFER/BUFLIST area.	369
READ_COSTATS	Retrieve statistical information for the specified cast-out classes.	389
READ_DATA	Read a data item from a data entry in the coupling facility cache structure into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA.	407
READ_DIRINFO	Retrieve directory information from the coupling facility cache structure and store it in the storage area specified by BUFFER or BUFLIST.	429
READ_STGSTATS	Retrieve statistical information for a specified storage class and store it in the storage area specified by STGSTATS.	449
REG_NAMELIST	Create a directory entry and register interest in a list of up to 32 data items and store the resulting state information in the storage area specified by NSBAREA.	459
RESET_REFBIT	Reset the reference bit in the directory entries for the specified cached data items to indicate that the data items were not recently referenced and obtain a count of referenced items.	475

Table 19. Request Types for IXLCACHE (continued)

Request Type	Description	Page
SET_RECLVCTR	Define and activate, or deactivate, a reclaiming vector for a specified storage class.	487
UNLOCK_CASTOUT	Releases the cast-out lock for the data items named in BUFFER or BUFLIST.	499
UNLOCK_CO_NAME	Releases the cast-out lock for a single data item named in CUNBAREA.	517
WRITE_DATA	Write data from the storage areas specified by BUFFER or BUFLIST and/or ADJAREA to a data entry in the coupling facility cache structure.	529
WRITE_DATALIST	Write data from a list of entries specified in either BUFFER or BUFLIST to data entries in the coupling facility cache structure.	555

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued.
AMODE:	31-bit
ASC mode:	Primary or access register (AR).
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Disabled callers must be legally disabled by holding the CPU lock and cannot hold other disabled locks. Enabled callers must not hold any locks. If MODE=SYNCSUSPEND is specified, the caller must be enabled.
Control parameters:	See “Restrictions” on page 247

Programming Requirements

- If your program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLCACHE. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.
- Include the IXLYCON mapping macro in your program. This macro provides a list of equate symbols for users of XES services and exits.
- Include mapping macros IXLYCAA, IXLYCANB, IXLYCCIH, IXLYCRRB, IXLYCSCS, IXLYCUNB, IXLYDEIB, IXLYNSB, IXLYDNNB, IXLYWOB, and IXLYWORB in your program as necessary. Table 20 lists these macros, the information and areas they map, and the particular IXLCACHE requests and parameters to which they apply.

Table 20. Mapping Macros for IXLCACHE

Mapping Macro	Information	Area	Request/Parameter
IXLYCAA	Answer area output	ANSAREA area	All requests
IXLYCANB	Entry name and associated directory information	BUFLIST buffers, BUFFER area	READ_DIRINFO (when DIRINFOFMT= NAMELIST), READ_COCLASS
IXLYCCIH	Cast-out class range	BUFLIST buffers, BUFFER area	READ_COSTATS
IXLYCRRB	Single registration block	BUFFER area	REG_NAMELIST
IXLYCSCS	Storage class statistics	STGSTATS area	READ_STGSTATS
IXLYCUNB	Entry name and associated control information	BUFLIST buffers, BUFFER area, CUNBAREA	UNLOCK_CASTOUT UNLOCK_CO_NAME

Table 20. Mapping Macros for IXLCACHE (continued)

Mapping Macro	Information	Area	Request/Parameter
IXLYDEIB	Directory entry	BUFLIST buffers, BUFFER area, DEIBAREA	READ_DIRINFO (when DIRINFOFMT= DIRENTRYLIST), READ_COCLASS (when DIRINFOFMT= DIRENTRYLIST), CASTOUT_DATALIST
IXLYDNNB	Entry name and optional comparative version	BUFLIST buffers, BUFFER area	DELETE_NAMELIST
IXLYNSB	Name and state information for registration blocks	NSBAREA	REG_NAMELIST
IXLYWOB	Entry name and associated control information.	BUFLIST buffers, BUFFER area	WRITE_DATALIST
IXLYWORB	Single write-operation-response block	WORBAREA	WRITE_DATALIST

Restrictions

- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).
- The caller's parameter list must be addressable in the caller's primary address space.
- If the caller is running in AR ASC mode and specifies a parameter using register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage areas specified by ADJAREA, ANSAREA, NSBAREA, and STGSTATS must be addressable in the caller's primary address space or from the caller's PASN access list. On a REG_NAMELIST request, the virtual storage area specified by BUFFER must be addressable in the caller's primary address space or from the caller's PASN access list.
- The virtual storage areas specified by parameters other than ADJAREA, ANSAREA, NSBAREA, and STGSTATS (and BUFFER on a REG_NAMELIST request) can be addressable in the caller's primary, secondary, or home address spaces, from the PASN access list, or from the dispatchable unit access list (DU-AL).
- If the caller is disabled, then the parameter list and all storage areas addressed by the parameters must reside in either nonpageable or disabled reference storage.
- If you specify BUFLIST and PAGEABLE=YES, all the buffers in the list must be in the same area of storage; you cannot mix common and private storage in the list of buffers.

Input Register Information

Before issuing the IXLCACHE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code
1	Used as work register
2-13	Unchanged
14	Used as work register
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work register
2-13	Unchanged
14-15	Used as work register

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLCACHE Version Support

The IXLCACHE macro supports versions in the range of 0 - 4.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLCACHE macro.
- The following keywords and functions are supported by all versions starting with version 1 and higher of the IXLCACHE macro.

ENDINDEX
NSBAREA
STARTINDEX

- The following keywords and functions are supported by all versions starting with version 2 and higher of the IXLCACHE macro.

CUNBAREA
RETURNDATA
REQUEST=UNLOCK_CO_NAME

- The following keywords and functions are supported by all versions starting with version 3 and higher of the IXLCACHE macro.

COSTATSFMT
DELETETYPE
ERRORACTION
NEWVERS
VERSCOMP
VERSCOMPTYPE
VERSUPDATE
REQUEST=DELETE_NAMELIST

- The following keyword is supported by all versions starting with version 4 and higher of the IXLCACHE macro. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

EXTRESTOKEN

- The following keywords and functions are supported by all versions starting with version 6 and higher of the IXLCACHE macro.

CASTOUTLIST
DATAOFFSET
DEIBAREA
WORBAREA
REQUEST=CASTOUT_DATALIST
REQUEST=CROSS_INVALLIST
REQUEST=WRITE_DATALIST

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

IXLCACHE REQUEST=CASTOUT_DATA

Description

A CASTOUT_DATA request allows you to read a data item from an entry in the cache structure to the storage areas specified by BUFFER or BUFLIST and/or ADJAREA. From these areas, the data can be written to permanent storage such as DASD. This is part of the cast-out process.

When you issue a CASTOUT_DATA request:

- A cast-out lock is obtained for the data item specified by NAME. This lock protects the data item from further cast-out processing until the cast-out lock is released. (To release the cast-out lock, you must issue an UNLOCK_CASTOUT request.)
- The change indicator in the directory entry for the data item is updated to indicate the data entry contains unchanged data. (Note that the data is considered changed until the cast-out lock is released.)

This request also allows you to register your interest (REGUSER=YES) in the data item. To register or update your interest in a data item, you must specify an index into your local cache vector (VECTORINDEX) to be associated with the named data item in the structure. This index identifies a vector entry that cache services uses to indicate both your interest in the data item and the validity of your locally cached copy of the data item. Having registered interest in the data item **means** that you have a valid copy of the data item in your local cache buffer. When another user updates the data item in the structure, cache services will update the data item's associated vector entry, thereby deregistering your interest in the data item and invalidating your locally cached copy.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=CASTOUT_DATA is as follows:

IXLCACHE Macro

main diagram

```
►►──IXLCACHE──b──REQUEST=CASTOUT_DATA──────────────────────────────────────────────────────►


     ┌─,OLDNAME=NO_OLDNAME──────────────────────────┐
  ┌─,REGUSER=NO─┼──────────────────────────────────────────────┼──────┐
  │       └─,OLDNAME=oldname─,VECTORINDEX=vectorindex─┘       │ ┌─,PROCESSID=NO_PROCESSID─┐
►─┼───────────────────────────────────────────────────────────────────┼─┼──────────────────────────┼──►
  │         ┌─,OLDNAME=NO_OLDNAME─┐                                  │ └─,PROCESSID=processid─────┘
  └─,REGUSER=YES─┼─────────────────────┼─,VECTORINDEX=vectorindex────┘
            └─,OLDNAME=oldname─────┘

                         ┌─,REQID=NO_REQID─┐
►─,CONTOKEN=contoken─────┼─────────────────┼──,NAME=name─────────────────────────────────────►
                         └─,REQID=reqid────┘

                                                            ┌─,ADJAREA=NO_ADJAREA─┐
►─┬─,BUFLIST=buflist─┤ parameters-1 ├───────────────────┬──┼─────────────────────┼──────────►
  └─,BUFFER=buffer─┤ parameters-2 ├─,BUFSIZE=bufsize────┘  └─,ADJAREA=adjarea────┘

  ┌─,MODE=SYNCSUSPEND──────────────────────────────┐   ┌─,ANSAREA=NO_ANSAREA──────────────┐
►─┼────────────────────────────────────────────────┼───┼──────────────────────────────────┼──►
  ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤   └─,ANSAREA=ansarea─,ANSLEN=anslen──┘
  │                 ┌─,REQDATA=NO_REQDATA─┐        │
  ├─,MODE=SYNCEXIT──┼─────────────────────┼────────┤
  │                 └─,REQDATA=reqdata────┘        │
  ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
  ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
  │                  ┌─,REQDATA=NO_REQDATA─┐       │
  ├─,MODE=ASYNCEXIT──┼─────────────────────┼───────┤
  │                  └─,REQDATA=reqdata────┘       │
  └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┘

                                                  ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬─────────────────────┬──┬─────────────────────┬─┼───────────────────────────┼──────────────►
  └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘ ├─,PLISTVER=MAX─────────────┤
                                                  └─,PLISTVER=plistver────────┘

  ┌─,MF=S───────────────────────────────────┐
►─┼─────────────────────────────────────────┼───────────────────────────────────────────────►◄
  │                     ┌─,0D──────┐        │
  ├─,MF=(L─,mfctrl──────┼──────────┼─)──────┤
  │                     └─,mfattr──┘        │
  │                     ┌─,COMPLETE─┐       │
  └─,MF=(E─,mfctrl──────┼───────────┼─)─────┘
                        └─,COMPLETE─┘
```

parameters-1

```
  ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─────────┐
├─┼──────────────────────────────────────────────────────────────────────────────────────┼──►
  │                                                ┌─,BUFALET=NO_BUFALET─┐ ┌─,BUFADDRSIZE=31─┐
  ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├──────────┼─────────────────────┼─┼─────────────────┼─
  │                                                └─,BUFALET=bufalet────┘ └─,BUFADDRSIZE=64─┘
```

Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=CASTOUT_DATA are listed in alphabetical order. Default values are underlined:

REQUEST=CASTOUT_DATA

Use this input parameter to:

- Specify that a cast-out lock be obtained for the data item identified by NAME, and that the named data item be read into the storage areas specified BUFFER or BUFLIST and/or ADJAREA.
- Optionally register interest in the data item (Specify REGUSER=YES also).

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*

Use this output parameter to specify a storage area to contain the adjunct data that was read from the designated entry.

If the structure supports adjunct data and ADJAREA is not specified, or if ADJAREA=NO_ADJAREA is specified, then no adjunct data is returned.

If the structure does not support adjunct data and ADJAREA is specified, then no adjunct data is returned, and the request will complete with a return code X'4' and a reason code of IXLRSNCODENOADJUNCTDATA.

| If the structure supports adjunct data and ADJAREA is specified, but there is no data and adjunct
| associated with the entry, then:

| - For READ_DATA requests, the request completes with return code X'4' and reason code
| IXLRSNCODENOREADDATA.
| - For CASTOUT_DATA requests, the request completes with return code X'8' and reason code
| IXLRSNCODECOUNCHANGED.

(Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte virtual storage area that the adjunct data will be read into.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro. On a successful completion of the request, the following information is returned in the answer area:

- The parity of the returned data item (field CAAPARITY).
- If REGUSER=YES is specified and the request replaced a previously specified local cache vector index for the entry, the replaced index is returned (fields CAAINVLCVI and CAAINVLCVINUM).
- If BUFFER or BUFLIST is specified, the number of elements in the retrieved entry is returned (field CAAELEMNUM).

To Code: Specify the RS-type name or address (using a register from 2 - 12) of an area (with a length of ANSLEN) where the information returned from the request will go.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the entry data that is cast out from the cache structure.

You can define the buffer size to be a total of up to 65536 bytes. Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to receive the entry requested.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold the entry data that is cast out of the cache structure. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. XES treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller’s PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously

- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=*name*
Use this input parameter to specify the name of the data item to be cast-out and for which the cast-out lock will be obtained.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,OLDNAME=NO_OLDNAME
,OLDNAME=*oldname*
Use this input parameter to specify the name of the data item for which your interest should be deregistered.

- For structures allocated in a coupling facility of CFLEVEL=4 or lower, you must also specify the name of the data item for which interest is to be registered after the interest is deregistered in OLDNAME. Identify the data item to which interest is to be registered with NAME. The VECTORINDEX currently associated with the entry specified by OLDNAME will be reassigned to the name of the data item specified by NAME.
- For structures allocated in a coupling facility of CFLEVEL=5 or higher, you can specify that interest in the name of the data item specified by OLDNAME is to be deregistered without registering interest in the entry specified by NAME.

In either case, whenever deregistration of interest is requested, VECTORINDEX must be specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the old data item that was associated with VECTORINDEX.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,PROCESSID=NO_PROCESSID
,PROCESSID=*processid*

Use this input parameter to specify a user-defined process identifier to be compared with the cast-out lock along with the connection identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the user-defined process identifier.

,REGUSER=NO
,REGUSER=YES

Use this input parameter to specify whether the request should register (or update) interest in the data item to be cast-out.

NO
Connection interest in the data item will not be registered.

For structures allocated in a coupling facility with CFLEVEL=5 or higher, OLDNAME may be specified to deregister any outstanding interest for the specified local cache vector index for a different entry. If OLDNAME is specified, then VECTORINDEX is required.

YES
Connection interest in the data item will be registered for the connection specified by CONTOKEN. If interest is already registered, the vector index specified by VECTORINDEX replaces any previously specified index for the data item.

Specify OLDNAME to deregister any outstanding interest for the specified vector index for a different entry.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VECTORINDEX=*vectorindex*
Use this input parameter to specify an index into your local cache vector to be associated with the data item specified by NAME, to be disassociated with the data item specified by OLDNAME, or both. The vector index identifies a vector entry that cache services will use to indicate both your interest in the data item and the validity of the copy of the data item in your local cache buffer.

If the vector index you specify is already associated with another data item, you must disassociate the vector index from the old name by specifying OLDNAME before the vector index can be associated with the data item specified by NAME.

On CASTOUT_DATA requests, for structures allocated in a coupling facility of CFLEVEL=5 or higher, VECTORINDEX is required when REGUSER=YES or OLDNAME is specified. For structures allocated in a coupling facility of CFLEVEL=4 or lower, VECTORINDEX is required when REGUSER=YES is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the vector index for NAME or OLDNAME.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx040C	**Equate Symbol:** IXLRSNCODENOADJUNCTDATA **Meaning:** The request specified that adjunct data was to be read, but the structure does not support adjunct areas. No adjunct data was retrieved; however, entry data was returned in BUFLIST or BUFFER if requested. The following fields were returned in the answer area: • CAAELEMNUM • CAAINVLCVI • CAAINVLCVINUM • CAAPARITY • CAAVERSION (for CFLEVEL=5 or higher) **Action:** No action is necessary. However, if you expected this structure to have adjunct data, you may want to: • Verify the CONTOKEN is from the correct invocation of IXLCONN. • Check the IXLCONN macro that defined the structure to be sure adjunct data was specified.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the specified storage area for adjunct data (ADJAREA) is not addressable. The adjunct data was not retrieved; however, the entry data was returned in BUFLIST or BUFFER, if requested. The following fields are returned in the answer area: • CAAELEMNUM • CAAINVLCVI • CAAINVLCVINUM • CAAPARITY For CFLEVEL=4 and higher coupling facilities, the following additional answer area field is returned: • CAADATACACHED **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro. Correct the address specified by ADJAREA and rerun the request asking for adjunct data only.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The VECTORINDEX specified is not valid. Request processing was suppressed. **Action:** Specify a vector index that is within the current range of the number of vector entries for the local vector requested for this structure. The number of vector entries is determined by the VECTORLEN parameter specified on the IXLCONN macro and may be changed by issuing the IXLVECTR macro.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because the entry designated by NAME is not in the cache structure. **Action:** If this is unexpected, try the following: • Verify that NAME is uncorrupted. • Verify that the CONTOKEN is for the correct structure. • Issue IXLCACHE REQUEST=READ_COCLASS to get the names of all the data items in this cast-out class. • Issue IXLCACHE REQUEST=READ_DIRINFO to get the cast-out class this name belongs in.
8	xxxx0827	**Equate Symbol:** IXLRSNCODECOLOCKHELD **Meaning:** Program error. The cast-out lock for NAME is already held. This request did not lock the entry for cast-out. No data was returned in BUFFER, BUFLIST, or ADJAREA. **Action:** A cast-out is already in progress for this data. The answer area (ANSAREA) contains the connection ID and the process ID in the CAACOLOCKVAL field of the user/process that currently holds the cast-out lock. The state of the cast-out lock is returned in the CAACOLOCKSTATE field. Check these values to determine if a problem exists in your code.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0828	**Equate Symbol:** IXLRSNCODECOUNCHANGED **Meaning:** Program error. The entry specified for cast-out does not contain changed data. The entry was not cast out, and no data was returned in BUFFER, BUFLIST, or ADJAREA. **Action:** A cast-out may not be done on unchanged data. (Unchanged indicates that the copy in the coupling facility cache structure is the same as the permanent storage copy.) Determine why you thought this entry should be cast-out (a WRITE_DATA with CHANGED=YES must have been done to indicate changed status). You could try one of the following to get more information about the cast-out class: • Issue IXLCACHE REQUEST=READ_COCLASS to get the names of all the data items in the cast-out class. • Issue IXLCACHE REQUEST=READ_DIRINFO to more information about changed data items.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program Error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data being read. The number of elements in the retrieved entry is returned in the answer area in the field CAAELEMNUM. **Action:** If more space is available, specify a larger buffer size and reissue the request.

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 21. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=CASTOUT_DATALIST

Description

A CASTOUT_DATALIST request allows you to read a set of data items from entries in the cache structure to the storage areas specified by BUFFER, BUFLIST, DEIBAREA, and/or ADJAREA. From these areas, the data can be written to permanent storage such as DASD. This is part of the cast-out process. The set of entries identified for castout processing is specified by CASTOUTLIST, which contains a list of up to 8 16-byte entry names.

The system references the entries in the CASTOUTLIST by an index into the list and processes them sequentially beginning with the entry specified by the first index (STARTINDEX) and ending with the entry identified by the last index (ENDINDEX). The entries are numbered starting with 1.

When you issue a CASTOUT_DATALIST request:

- A cast-out lock is obtained for the set of entries specified in CASTOUTLIST. This lock protects the data item from further cast-out processing until the cast-out lock is released. (To release the cast-out lock, you must issue an UNLOCK_CASTOUT request.)
- The change indicator in the directory entry for each entry is updated to indicate the data entry contains unchanged data. (Note that the data is considered changed until the cast-out lock is released.)

When processing is initiated for a CASTOUT_DATALIST request, the system moves the DEIB, which contains directory information for the entry, to DEIBAREA and adjunct data, if present in the structure, to ADJAREA for the first processed name in the CASTOUTLIST pointed to by STARTINDEX. The data area from the first processed entry is moved to the output BUFFER or BUFLIST. Subsequent processing of entries in the CASTOUTLIST move the DEIB, adjunct data if it exists, and the data area from the processed entry to the output BUFFER or BUFLIST. This continues until the end of the CASTOUTLIST pointed to by ENDINDEX is reached or until processing is discontinued.

Processing for a CASTOUT_DATALIST request can be discontinued in the following instances:

- Another connection or process currently holds the cast-out lock for the entry being processed. The entry is not processed and ANSAREA contains:
  - The index of the name in the CASTOUTLIST
  - The value of the cast-out lock
  - The value of the cast-out lock state

  All prior entries in the CASTOUTLIST will have been processed.
- Entry data is not cached, or the cached entry data is not changed. The entry is not processed and ANSAREA contains:
  - The index of the name in the CASTOUTLIST
  - The changed data indicator
  - The data-cached indicator

  All prior entries in the CASTOUTLIST will have been processed.
- The entry name being processed in the CASTOUTLIST is not in the directory. The entry is not processed and ANSAREA contains:
  - The index of the name in the CASTOUTLIST

  All prior entries in the CASTOUTLIST will have been processed.
- The size of the BUFFER or BUFLIST is not large enough to contain the data area for the entry specified by STARTINDEX. The entry is not processed and ANSAREA contains:
  - The number of elements in the indexed entry

No entries are processed.

- The remaining space in the BUFFER or BUFLIST is not large enough to contain the data area for the current entry in the CASTOUTLIST. The entry is not processed and ANSAREA contains:
  - The index of the entry in CASTOUTLIST being processed
  - The number of elements in the desired entry

  All prior entries in the CASTOUTLIST will have been processed.

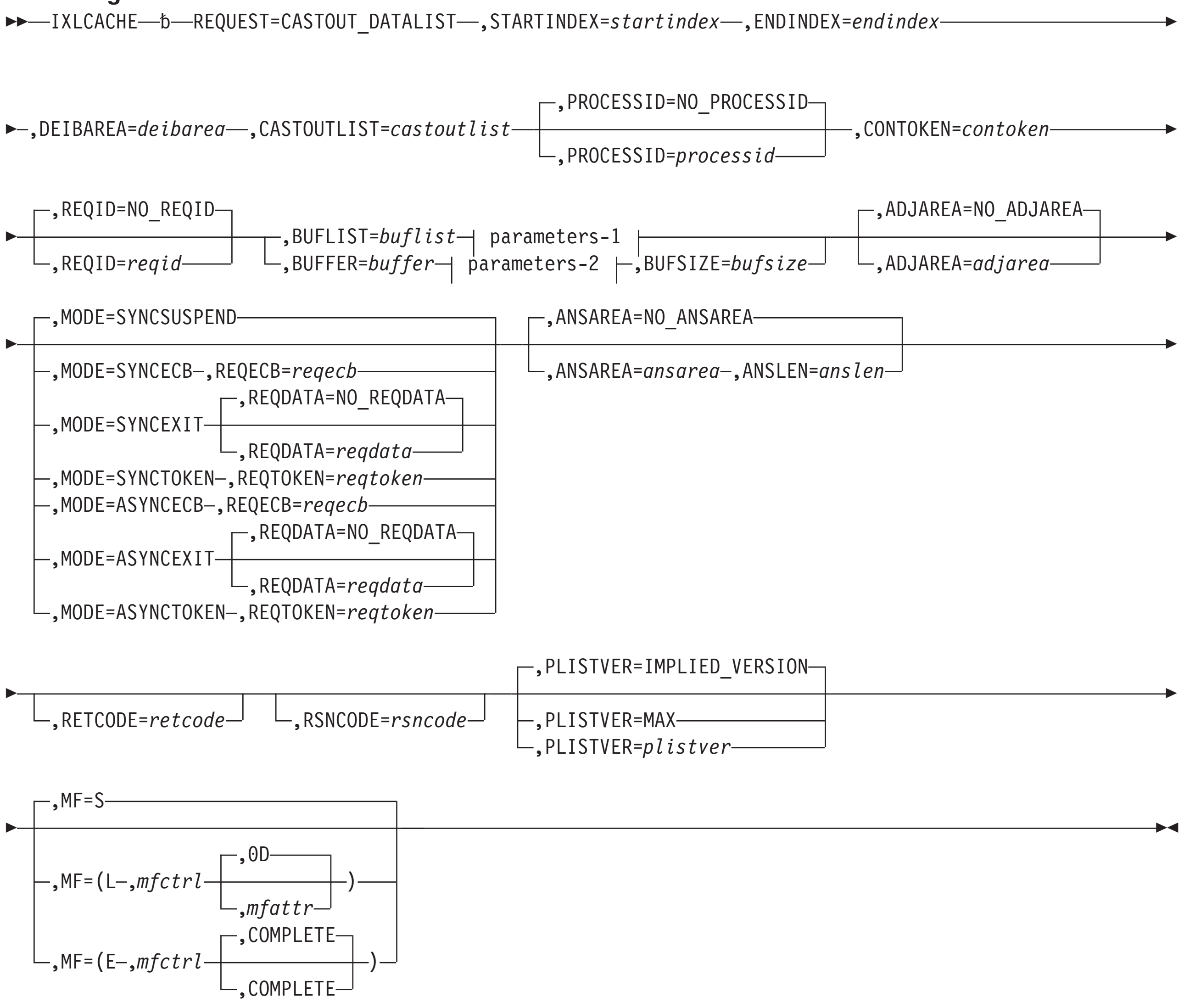
A CASTOUT_DATALIST can complete prematurely because the request exceeds the time-out criteria for the coupling facility. (Time-out criteria is model-dependent.) When a request completes prematurely, the system returns an index value (CAACDLINDEX) in the answer area that you can use to restart the CASTOUT_DATALIST request. The index value when a CASTOUT_DATALIST completes prematurely is the index of the first unprocessed entry in the CASTOUTLIST.

A CASTOUT_DATALIST request can be issued only for cache structures allocated in a coupling facility of CFLEVEL=12 or higher. CASTOUT_DATALIST requests issued for a cache structure allocated in a coupling facility of CFLEVEL=0 through 11 will fail.

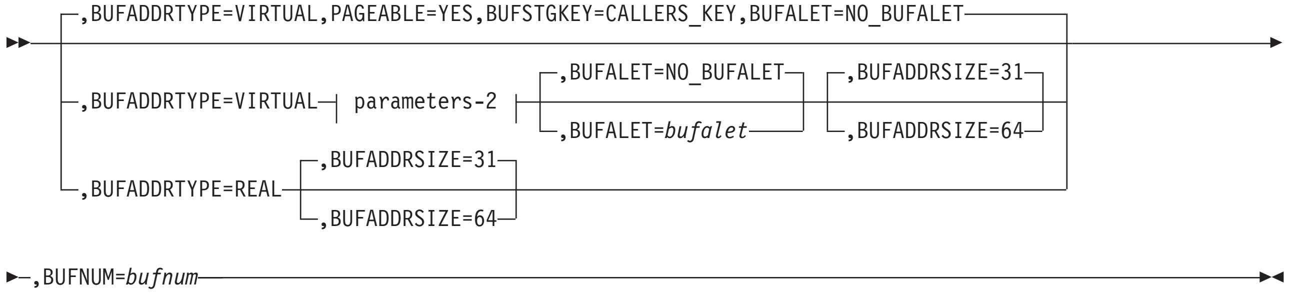
Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=CASTOUT_DATALIST is as follows:

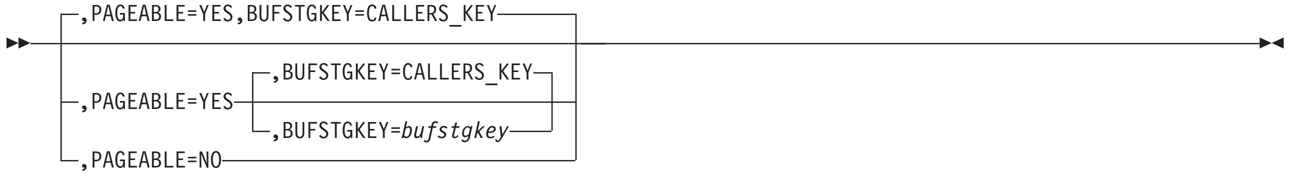
main diagram



parameters-1



parameters-2



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=CASTOUT_DATALIST are listed in alphabetical order. Default values are underlined:

REQUEST=CASTOUT_DATALIST

Use this input parameter to:

- Specify that a cast-out lock be obtained for the set of data items identified by CASTOUTLIST, and that the named data items be read into the storage areas specified BUFFER or BUFLIST, DEIBAREA, and/or ADJAREA.
- Update the directory entry change bit for each data item to indicate that each entry contains unchanged subsystem data.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*

Use this output parameter to specify a storage area to contain the adjunct data that was read from the first processed entry.

If the structure supports adjunct data and ADJAREA is not specified, or if ADJAREA=NO_ADJAREA is specified, then no adjunct data is returned for the first entry processed.

If the structure does not support adjunct data and ADJAREA is specified, then entry data is returned, but no adjunct data is returned for every entry processed. CAAADJAREAVALID, which is returned in ANSAREA, will indicate that adjunct data did not exist.

(Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte virtual storage area that the adjunct data will be read into.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 - 12) of an area (with a length of ANSLEN) where the information returned from the request will go.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL

,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL

The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL

The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the data area from the first processed name in the CASTOUTLIST followed by the DEIB, adjunct data if it exists, and the data area from the second processed name in the CASTOUTLIST. This will continue until the end of the CASTOUTLIST pointed to by ENDINDEX is reached or until the request completes prematurely for some other reason.

You must ensure that the storage area specified by BUFFER:

- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.

See the BUFSIZE parameter description for defining the size of the buffer.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to receive the entries requested.

BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold the data area from the first processed name in the CASTOUTLIST followed by the DEIB, adjunct data if it exists, and the data area from the second processed name in the CASTOUTLIST. This will continue until the end of the CASTOUTLIST pointed to by ENDINDEX is reached or until the request completes prematurely for some other reason. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. Cache services treat BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM parameter to specify the number of buffers in the buffer list.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CASTOUTLIST=*castoutlist*
Use this input parameter to specify a 128-byte area that contains a list of up to 8 16-byte entry names to be locked for castout processing.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field containing a list of entry names.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DEIBAREA=*deibarea*
Use this output parameter to specify an output area to contain the DEIB from the first entry processed in CASTOUTLIST, as specified by STARTINDEX. The DEIBs for the rest of the entries will be returned in the data buffer specified by BUFFER or BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field to contain the DEIB from the first entry processed.

,ENDINDEX=*endindex*
Use this input parameter to specify the ending index for the last entry in CASTOUTLIST to be processed. The index value must be greater than or equal to the value specified for STARTINDEX, but less than or equal to 8.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the ending index value.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,PROCESSID=NO_PROCESSID
,PROCESSID=*processid*
Use this input parameter to specify a user-defined process identifier to be compared with the cast-out lock along with the connection identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the user-defined process identifier.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be

processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*
Use this input parameter to specify the index into CASTOUTLIST for the first entry to be processed. Valid STARTINDEX values are from 1 to the value of ENDINDEX. The first name in CASTOUTLIST block has index number 1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the starting index value.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the first unprocessed entry in the CASTOUTLIST has been returned in the answer area (field CAACDLINDEX). All prior entries have been processed. **Action:** To restart the request, update STARTINDEX with the value of CAACDLINDEX, the index of the next entry in CASTOUTLIST to be processed. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the specified storage area for adjunct data (ADJAREA) is not addressable. The adjunct data was not retrieved; however, the entry data was returned in BUFLIST or BUFFER, if requested. The following fields are returned in the answer area: • CAAELEMNUM • CAAINVLCVI • CAAINVLCVINUM • CAAPARITY For CFLEVEL=4 and higher coupling facilities, the following additional answer area field is returned: • CAADATACACHED **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro. Correct the address specified by ADJAREA and rerun the request asking for adjunct data only.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** The request completed prematurely because the buffer specified by BUFFER, or the buffers specified by BUFLIST, is full. The index of the name in the CASTOUTLIST returned in the answer area (field CAACDLINDEX) contains the index of the entry currently being processed and the number of elements in the desired entry is also returned in the answer area (field CAAELEMNUM). **Action:** If ADJAREA is specified, check the CAAADJAREAVALID bit in the answer area to see if adjunct was returned. If ADJAREA was specified, check the CAAADJAREANONADDR bit to determine if the provided storage for the adjunct data is addressable. If the provided storage for the DEIBAREA is not addressable, then the CAADEIBAREANONADDR bit will be set. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because a name element specified an entry name that did not exist in the cache structure. The index of the failing entry is returned and all prior names were processed. **Action:** If this is unexpected, try the following: • Verify that entry name is uncorrupted. • Verify that the CONTOKEN is for the correct structure.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	**Equate Symbol:** IXLRSNCODECOLOCKHELD **Meaning:** Program error. The cast-out lock for the entry name currently being processed in the CASTOUTLIST is already held. This request did not lock the entry for cast-out. The index of the name in the CASTOUTLIST, the value of the cast-out lock (CAACOLOCKVAL), and the value of the cast-out lock state (CAACOLOCKSTATE) are returned in the answer area. All prior entries in the CASTOUTLIST were processed. **Action:** If ADJAREA was specified, check the CAAADJAREAVALID bit in the answer area to see if adjunct was returned. If ADJAREA was specified, check the CAAADJAREANONADDR bit to determine if the provided storage for the adjunct data is addressable. If the provided storage for the DEIBAREA is not addressable, then the CAADEIBAREANONADDR bit will be set.
8	xxxx0828	**Equate Symbol:** IXLRSNCODECOUNCHANGED **Meaning:** Program error. The entry specified for cast-out does not contain changed data. The entry was not cast out. The system returns the index of the name in the CASTOUTLIST, the changed indicator, and the data cached indicator in the answer area. All prior entries in the CASTOUTLIST were processed. **Action:** If ADJAREA was specified, check the CAAADJAREAVALID bit in the answer area to see if adjunct was returned. If ADJAREA was specified, check the CAAADJAREANONADDR bit to determine if the provided storage for the adjunct data is addressable. If the provided storage for the DEIBAREA is not addressable, then the CAADEIBAREANONADDR bit will be set.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The name index specified by either STARTINDEX or ENDINDEX is not valid. No elements have been processed. **Action:** Ensure that STARTINDEX and ENDINDEX specify valid indexes into the name elements stored in CASTOUTLIST. The value of ENDINDEX must be greater than or equal to STARTINDEX and ENDINDEX must be less than or equal to 8.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program Error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data area for the entry in the CASTOUTLIST specified by STARTINDEX. The number of elements in the retrieved entry is returned in the answer area in the field CAAELEMNUM. **Action:** If more space is available, specify a larger buffer size and reissue the request.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=12 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.

Table 22. Return and Reason Codes for the IXLCACHE REQUEST=CASTOUT_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=CROSS_INVAL

Description

A CROSS_INVAL request allows you to invalidate the copies of the data item in the local cache buffers of other users with registered interest in the data item. A cross-invalidate operation:

- Deregisters user interest. With the exception of your connection, all users with registered interest in the data item have their interest deregistered.
- Invalidates local copies of the data item. With the exception of the copy in your local cache buffer, all users' copies of the data item are invalidated.

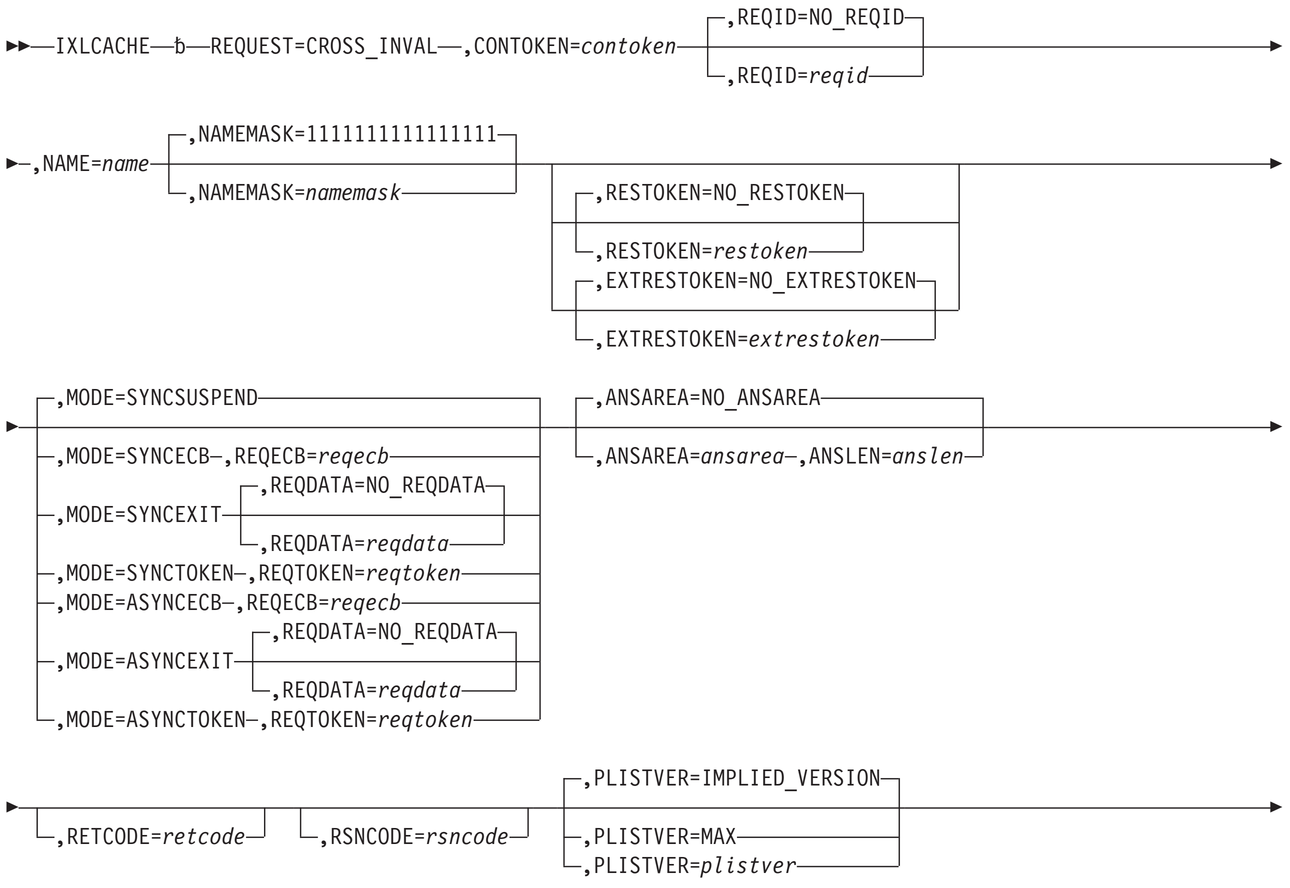
The data items against which a cross-invalidate operation is performed are identified by NAME and NAMEMASK.

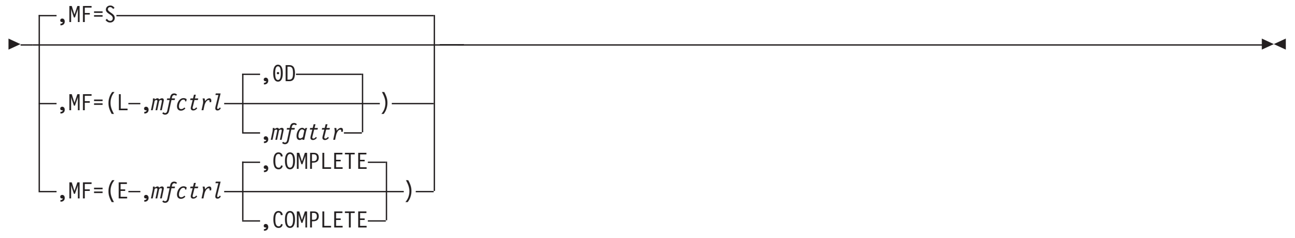
If the request exceeds the model-dependent time-out criteria before processing completes, either a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area. The token can be specified on the next CROSS_INVAL request to resume processing with the next data item to be processed. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=CROSS_INVAL is as follows:

main diagram





Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=CROSS_INVAL are listed in alphabetical order. Default values are underlined:

REQUEST=CROSS_INVAL
Use this input parameter to specify that a cross-invalidate operation be performed for one or more data items specified by NAME and NAMEMASK. All users except for your connection have their interest in the data item deregistered and the copies of the data item in their local cache buffers invalidated.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain a restart token (CAARESTOKEN) or extended restart token (CAAEXTRESTOKEN) that is returned from a request that exceeds the model-dependent time-out criteria. See the RESTOKEN and EXTRESTOKEN parameters for a description of the restart token.

The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,EXTRESTOKEN=<u>NO_EXTRESTOKEN</u>
,EXTRESTOKEN=*extrestoken*
Use this input parameter to specify an extended restart token that can be used to resume processing of a CROSS_INVAL request that exceeded the model-dependent time-out criteria. The extended restart token is returned in the answer area (field CAAEXTRESTOKEN), and should be specified on the next CROSS_INVAL request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria, the extended restart token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.
3. Specifying an extended restart token requires that the length of the answer area be at least the length of CAALEVEL1LEN.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND

,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=*name*
Use this input parameter to specify the name of the data item against which a cross-invalidate operation will be performed. With the exception of your connection, all users with registered interest in the data item will have their interest deregistered and the copies of the data item in their local cache buffers invalidated.

You may cross-invalidate multiple data items with one invocation of this macro, if they have similar names. See the description of the NAMEMASK parameter for how to do this.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NAMEMASK=1111111111111111

,NAMEMASK=*namemask*
Use this input parameter to specify which characters in the name specified by NAME are to be used in selecting data items for processing. This parameter allows you to select multiple data items based on common characters in NAME.

The position of each bit in NAMEMASK corresponds to the same relative character position in NAME. A one indicates that the corresponding letter should be used in selecting entries; a zero indicates that the corresponding letter should not be used.

Specifying a name mask with all zeros causes all names to be selected for processing. Specifying a name mask with all ones causes only the name specified by NAME to be selected.

For more information on how NAMEMASK may be used, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the bit-mask for the name specified on the NAME keyword.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be

processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*

Use this input parameter to specify a restart token that can be used to resume processing of a CROSS_INVAL request that exceeded the model-dependent time-out criteria. The restart token is returned in the answer area (field CAARESTOKEN), and should be specified on the next CROSS_INVAL request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria, the returned token will not be provided

Notes:

1. Specifying a restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify a restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING

8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. A token for restarting the request has been returned in the answer area (field CAARESTOKEN or CAAEXTRESTOKEN). **Action:** Reissue the request using the restart token (RESTOKEN or CAAEXTRESTOKEN). For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to resume the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request. • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. **Action:**Discard the results of the initial request and reissue the request with an EXTRESTOKEN value of zero. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 23. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVAL Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=CROSS_INVALLIST

Description

A CROSS_INVALLIST request allows you to invalidate the copies of a set of data items in the local cache buffers of other users with registered interest in the data item. A cross-invalidate operation:

- Deregisters user interest. With the exception of your connection, all users with registered interest in the data item have their interest deregistered.
- Invalidates local copies of the data item. With the exception of the copy in your local cache buffer, all users' copies of the data item are invalidated.

A CROSS_INVALLIST request allows you to specify that a cross-invalidate operation be performed against a set of 1 to 4096 entries identified by a list of names in the BUFFER storage area or the buffers specified by BUFLIST.

The system references the name elements by an index into the buffer and processes them sequentially beginning with the name block identify by the first index (STARTINDEX) and ending with the data item identified by the last index (ENDINDEX). The name elements are numbered starting with 1.

If any name element fails to identify an existing structure entry, processing either continues or terminates, depending on the value specified for the ERRORACTION keyword. If ERRORACTION=TERMINATE, processing stops and the index value of the name element that caused the problem is returned in the CAACILINDEX field in the IXLYCAA answer area. If ERRORACTION=CONTINUE, processing continues with the next name element.

A CROSS_INVALLIST can complete prematurely because the request exceeds the time-out criteria for the coupling facility. (Time-out criteria is model-dependent.) When a request completes prematurely, the system returns an index value (CAACILINDEX) in the answer area which you can use to restart the CROSS_INVALLIST request. The index value returned when a CROSS_INVALLIST request completes prematurely is the index of the first unprocessed name element.

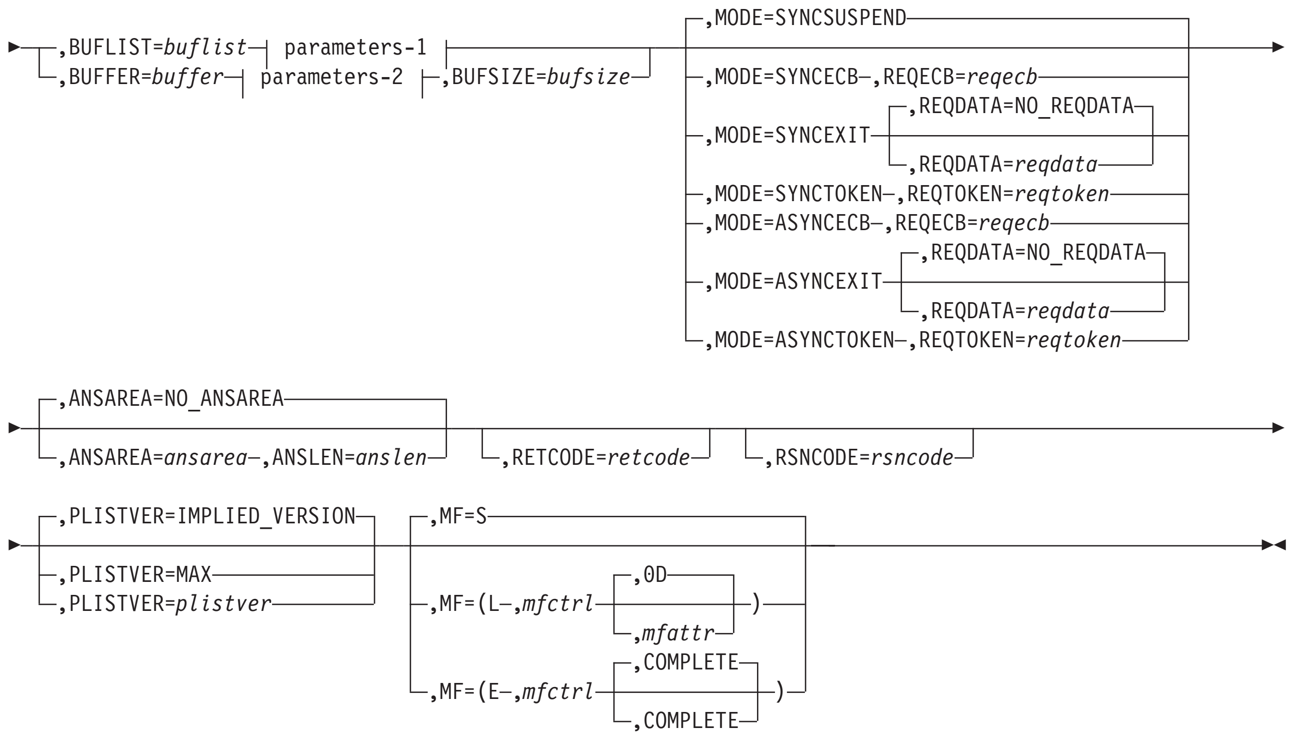
A CROSS_INVALLIST request can be issued only for cache structures allocated in a coupling facility of CFLEVEL=12 or higher. CROSS_INVALLIST requests issued for a cache structure allocated in a coupling facility of CFLEVEL=0 through 11 will fail.

Syntax Diagram

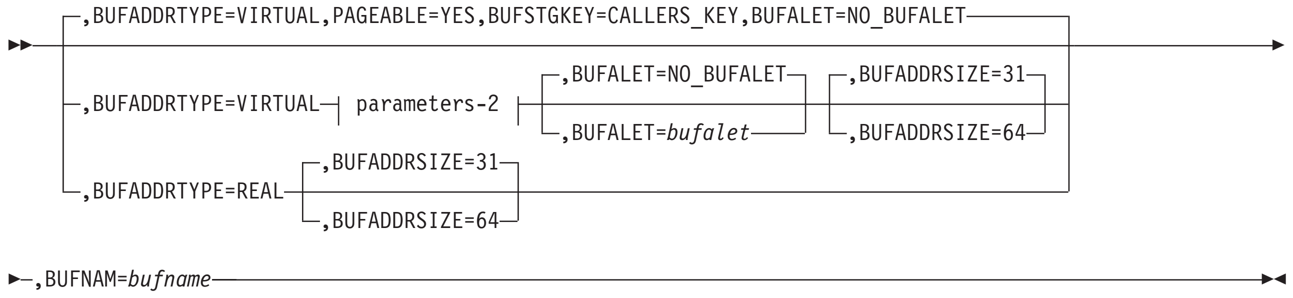
The syntax diagram for IXLCACHE REQUEST=CROSS_INVALLIST is as follows:

main diagram

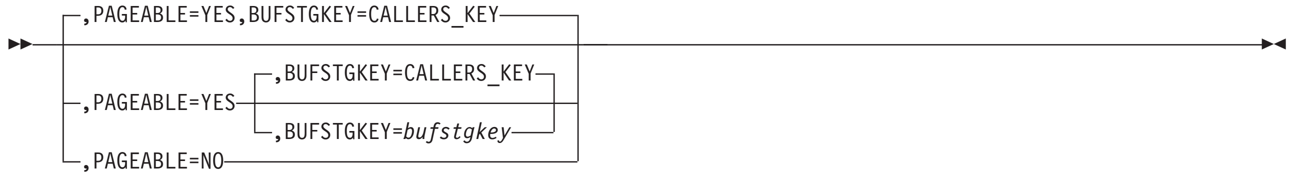
```
►►──IXLCACHE──␢──REQUEST=CROSS_INVALLIST──,STARTINDEX=startindex──,ENDINDEX=endindex──────────►

   ┌─,ERRORACTION=CONTINUE──┐                      ┌─,REQID=NO_REQID─┐
►──┼────────────────────────┼─,CONTOKEN=contoken───┼─────────────────┼──────────►
   └─,ERRORACTION=TERMINATE─┘                      └─,REQID=reqid────┘
```



parameters-1



parameters-2



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=CROSS_INVALLIST are listed in alphabetical order. Default values are underlined:

REQUEST=CROSS_INVALLIST
Use this input parameter to specify that a cross-invalidate operation be performed against a given set of 1 to 4096 entries specified by a list of names in the BUFFER storage area or the buffers specified

by BUFLIST. STARTINDEX and ENDINDEX determine the set of names provided by the user in the BUFFER or BUFLIST that will be processed by the request. Valid STARTINDEX and ENDINDEX values are 1-origin. The result of the operation is that with the exception of the connection specified by CONTOKEN, all connections that registered interest in the specified entries have interest deregistered and a cross-invalidate performed against their local caches.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request if the request does not complete successfully. See the return and reason codes descriptions for this request to determine which fields in the answer area are valid for non-zero return codes.

The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to contain the set of 16-byte structure entry names on which a cross-invalidate operation is to be performed.

Requirements for the buffer length are as follows:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer. See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the structure entry names.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold the structure entry names identifying the entries on which a cross-invalidate operation is to be performed. BUFLIST specifies a 128-byte storage area that consists of a list of 1 to 16 buffer address elements.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space as defined by BUFALET.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes as defined by BUFINCRNUM.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. XES treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a list of buffer addresses.

| **,BUFNUM=***bufnum*
| Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 1 to 16.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (1 to 16) in the list (BUFLIST)

| **,BUFSIZE=***bufsize*
| Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

| **,BUFSTGKEY=CALLERS_KEY**
| **,BUFSTGKEY=***bufstgkey*
| Use this input parameter to specify a storage key that you define and use when referencing the buffer specified by BUFFER.

| If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer are in the caller's PSW key.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage key in the format kkkkxxxx, where kkkk is the key and xxxx is ignored.

| **,CONTOKEN=***contoken*
| Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

| **,ENDINDEX=***endindex*
| Use this input parameter to specify the index into the name elements in the buffer storage area or the buffers specified by BUFLIST of the last name element to be processed. The index value must be greater than or equal to the value specified for STARTINDEX.

| **To Code:** Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the ending index value.

| **ERRORACTION=CONTINUE**
| **ERRORACTION=TERMINATE**
| Use this input parameter to specify whether processing is to continue with the next name element if an entry is not found.

| **CONTINUE**
| If an error occurs, processing is to continue with the next name element.

| **TERMINATE**
| If an error occurs, processing is to halt and the index of the name element that caused the error is returned in the answer area.

| **,MF=S**
| **,MF=(L,***mfctrl***)**
| **,MF=(L,***mfctrl,mfattr***)**
| **,MF=(L,***mfctrl***,0D)**
| **,MF=(E,***mfctrl***)**

,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
: Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
: Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
: Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*
: Use this input parameter to specify the starting index for name element processing. Valid STARTINDEX values are from 1 to the value of ENDINDEX. The first name element has index number 1.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the starting index value.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the first unprocessed entry has been returned in the answer area (field CAACILINDEX). All prior entries have been processed. **Action:** To restart the request, update STARTINDEX with the value of CAACILINDEX, the index of the next entry in the list of names to be processed in the BUFFER storage area or the buffers specified by BUFLIST. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because a name element specified an entry name that does not exist in the cache structure and ERRORACTION=TERMINATE was specified. The index of the failing name is returned in the answer area (field CAACILINDEX). All prior name elements were processed. **Action:** Reissue the request specifying for STARTINDEX the index of the next valid name element to be processed.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The name index specified by either STARTINDEX or ENDINDEX is not valid. No elements have been processed. **Action:** Ensure that STARTINDEX and ENDINDEX specify valid indexes into the name elements stored in BUFLIST or BUFFER. The value of ENDINDEX must be greater than or equal to STARTINDEX and ENDINDEX must be less than or equal to 4096.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • Requirements for BUFFER and BUFSIZE. • Buffer boundaries.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=12 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.

Table 24. Return and Reason Codes for the IXLCACHE REQUEST=CROSS_INVALLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=DELETE_NAME

Description

A DELETE_NAME request allows you to delete one or more data items identified by NAME and NAMEMASK from the cache structure. For cache structures allocated in a coupling facility of CFLEVEL=4 or lower, a DELETE_NAME request frees directory and data-entry resources for reuse. All users with registered interest in the data items have their interest deregistered and the copies of the data items in their local cache buffers invalidated. The data items are also removed from storage and cast-out classes.

For cache structures allocated in a coupling facility of CFLEVEL=5 or higher, you can specify for each data entry what resources are to be deleted. The DELETETYPE keyword provides options that allow you to delete the directory entry and associated data, only the changed data but not the directory entry, only the unchanged data but not the directory data, or any data (changed or unchanged) but not the directory entry.

Optionally, with a structure allocated in a coupling facility of CFLEVEL=5 or higher, you can specify that a version comparison is to be made for each entry processed. If the comparison does not meet the condition specified for the comparison, the entry is not processed and processing continues with the next entry to be considered.

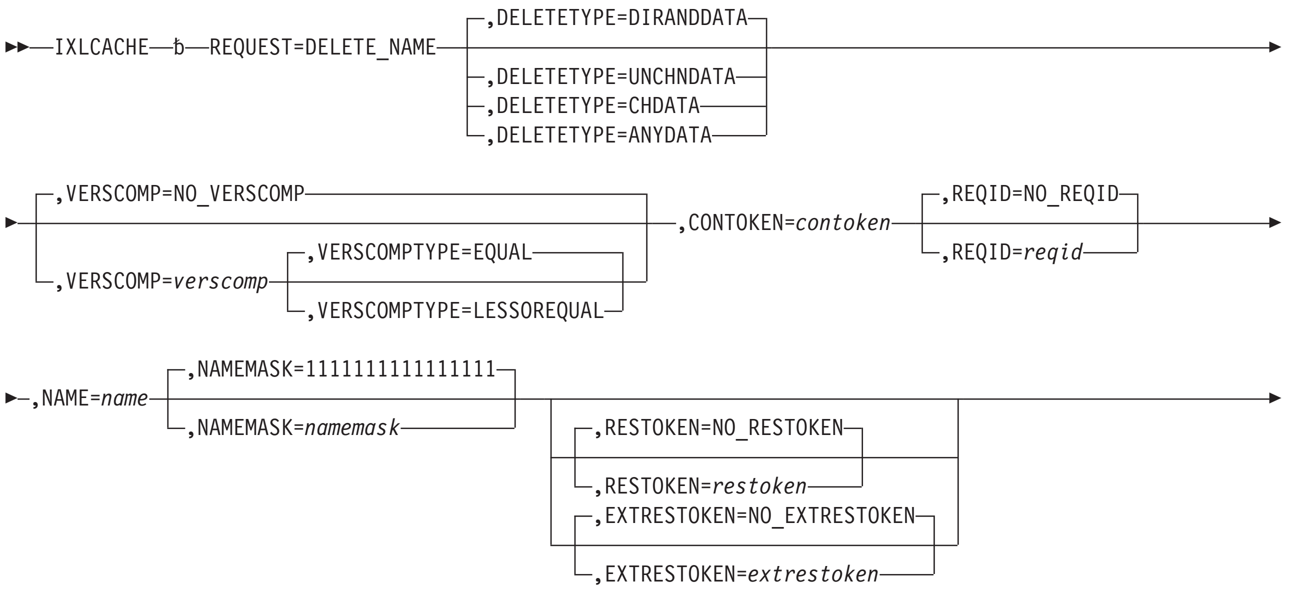
In a coupling facility of CFLEVEL=7 or higher, you can use NAMEMASK on an IXLCACHE DELETE_NAME request in conjunction with NAMECLASSMASK (specified on the IXLCONN request to connect to the cache structure) to improve the efficiency of selecting data items for processing.

If the request exceeds the model-dependent time-out criteria before it completes, a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area (mapped by IXLYCAA). The token can be specified on the next DELETE_NAME request to resume processing with the next data item to be deleted. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

The syntax diagram for the IXLCACHE REQUEST=DELETE_NAME is as follows:

main diagram



```
   ┌─,MODE=SYNCSUSPEND──────────────────────────┐   ┌─,ANSAREA=NO_ANSAREA──────────────┐
►──┼────────────────────────────────────────────┼───┼──────────────────────────────────┼──────────────────────►
   ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤   └─,ANSAREA=ansarea─,ANSLEN=anslen──┘
   │               ┌─,REQDATA=NO_REQDATA─┐      │
   ├─,MODE=SYNCEXIT┼─────────────────────┼──────┤
   │               └─,REQDATA=reqdata────┘      │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
   │                ┌─,REQDATA=NO_REQDATA─┐     │
   ├─,MODE=ASYNCEXIT┼─────────────────────┼─────┤
   │                └─,REQDATA=reqdata────┘     │
   └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┘

                                                  ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────────┬──┬─────────────────────┬──┼───────────────────────────┼──────────────────────────►
   └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘  ├─,PLISTVER=MAX─────────────┤
                                                  └─,PLISTVER=plistver────────┘

   ┌─,MF=S─────────────────────────────┐
►──┼───────────────────────────────────┼────────────────────────────────────────────────────────────────►◄
   │                 ┌─,0D──────┐      │
   ├─,MF=(L─,mfctrl──┼──────────┼─)────┤
   │                 └─,mfattr──┘      │
   │                 ┌─,COMPLETE─┐     │
   └─,MF=(E─,mfctrl──┼───────────┼─)───┘
                     └─,COMPLETE─┘
```

Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=DELETE_NAME are listed in alphabetical order. Default values are underlined:

REQUEST=DELETE_NAME
Use this input parameter to specify that one or more data items specified by NAME and NAMEMASK are to be deleted from the cache structure.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain a restart token or extended restart token that is returned from a request that exceeds the model-dependent time-out criteria. See the RESTOKEN and EXTRESTOKEN parameters for a description of the restart token.

The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DELETETYPE=DIRANDDATA
,DELETETYPE=UNCHNDATA
,DELETETYPE=CHDATA
,DELETETYPE=ANYDATA

Use this input parameter to specify the type of delete processing that is to be performed. DELETETYPE is valid only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

DIRANDDATA
For each applicable structure entry, invalidate the name and remove the name from storage and cast-out classes, and release all directory and data entry resources for the structure for reuse by the structure. The system deregisters interest in all connections with registered interest in the entry and performs a cross-invalidate against their local caches.

UNCHNDATA
For each applicable structure entry, if the data is unchanged, release the data entry resources for the entry for reuse by the structure. The system does not delete the directory entry and does not perform a cross-invalidate.

CHDATA
For each applicable structure, if the data is changed, release the data entry resources for the entry for reuse by the structure, set the change bit, the castout lock, and the castout lock state to zero, and remove the directory entry from the castout class. The system does not delete the directory entry and does not perform a cross-invalidate.

ANYDATA
For each applicable structure, whether the data is changed or unchanged, release the data entry resources for the entry for reuse by the structure. If the data is changed, set the change bit, the castout lock, and the castout lock state to zero, and remove the directory entry from the cast-out class. The system does not delete the directory entry and does not perform a cross-invalidate.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*

Use this input parameter to specify an extended restart token that can be used to resume processing of a DELETE_NAME request that exceeded the model-dependent time-out criteria. The extended restart token is returned in the answer area (field CAAEXTRESTOKEN), and should be specified on the next DELETE_NAME request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria, the extended restart token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.
3. Specifying an extended restart token requires that the length of the answer area be at least the length of CAALEVEL1LEN.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=*name*
Use this input parameter to specify the name of the data item to be deleted from the cache structure.

You may delete multiple data items with one invocation of this macro, if they have similar names. See the description of the NAMEMASK parameter for how to do this.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NAMEMASK=1111111111111111
,NAMEMASK=*namemask*
Use this input parameter to specify which characters in the name specified by NAME are to be used in selecting data items for processing. This parameter allows you to select multiple data items based on common characters in NAME.

The position of each bit in NAMEMASK corresponds to the same relative character position in NAME. A one indicates that the corresponding letter should be used in selecting entries; a zero indicates that the corresponding letter should not be used.

Specifying a name mask with all zeros causes all names to be selected for processing. Specifying a name mask with all ones causes only the name specified by NAME to be selected.

For more information on how NAMEMASK may be used, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the bit-mask for the name specified on the NAME keyword.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*

Use this input parameter to specify a restart token that can be used to resume processing of a DELETE_NAME request that exceeded the model-dependent time-out criteria. The restart token is returned in the answer area, and should be specified on the next DELETE_NAME request to resume processing with the next data item to be deleted.

If the request does not exceed the model-dependent time-out criteria, the restart token will not be provided.

Notes:

1. Specifying a restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify a restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*

Use this imput parameter to specify a value to be compared to the version number of the entry designated by NAME.

VERSCOMP is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

If the condition specified by VERSCOMPTYPE is not met, the entry is not processed and processing continues with the next entry to be considered.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains a value to be compared to each entry’s version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the structure entry version number comparison is to be performed.

VERSCOMPTYPE is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher and may be specified only when VERSCOMP also is specified.

VERSCOMPTYPE=EQUAL
The version number for the structure entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the structure entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

IXLCACHE Macro

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. A token for restarting the request has been returned in the answer area (field CAARESTOKEN or CAAEXTRESTOKEN). **Action:** Reissue the request using the restart token (RESTOKEN or CAAEXTRESTOKEN). For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to resume the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request. • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. **Action:**Discard the results of the initial request and reissue the request with an EXTRESTOKEN value of zero. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 25. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAME Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=DELETE_NAMELIST

Description

A DELETE_NAMELIST request allows you to specify a set of data entries that you want to delete from the structure. Each entry is represented by a name block, mapped by the IXLYDNNB macro. The name block contains the name of the structure entry for which delete processing is to be performed and optionally, a version comparison value used when version comparison is requested. The name blocks are contained in the storage area specified by BUFFER or the buffers specified by BUFLIST.

The system references the name blocks by an index into the buffer and processes them sequentially beginning with the name block identify by the first index (STARTINDEX) and ending with the data item identified by the last index (ENDINDEX). The name blocks are numbered starting with one.

For each structure entry, the DELETETYPE keyword indicates what specifically will be deleted. The DELETETYPE keyword provides options that allow you to delete the directory entry and associated data, only the changed data but not the directory entry, only the unchanged data but not the directory entry, or any entry (changed or unchanged) but not the directory data.

If any name block fails to identify an existing structure entry, processing either continues or terminates, depending on the value specified for the ERRORACTION keyword. If ERRORACTION=TERMINATE, processing stops and the index value of the name block that caused the problem is returned in the CAADNLINDEX field in the IXLYCAA answer area. If ERRORACTION=CONTINUE, processing continues with the next name block.

Each name block can optionally contain a comparative version number that is to be compared with the version number associated with the cache structure entry identified in the name block. The VERSCOMPTYPE keyword is used to define how the comparison is to be performed. If a version comparison mismatch occurs, then processing will either terminate or continue depending on the value specified for the ERRORACTION keyword. If ERRORACTION=TERMINATE is specified, processing stops and the index value of the name block that caused the problem is returned in the CAADNLINDEX field in the IXLYCAA answer area. If ERRORACTION=CONTINUE is specified, processing continues with the next name block.

A DELETE_NAMELIST can complete prematurely because the request exceeds the time-out criteria for the coupling facility. (Time-out criteria is model-dependent.) When a request completes prematurely, the system returns an index value (CAADNLINDEX) in the answer area which you can use to restart the DELETE_NAMELIST request. The index value returned when a DELETE_NAMELIST request completes prematurely is the index of the first unprocessed name block.

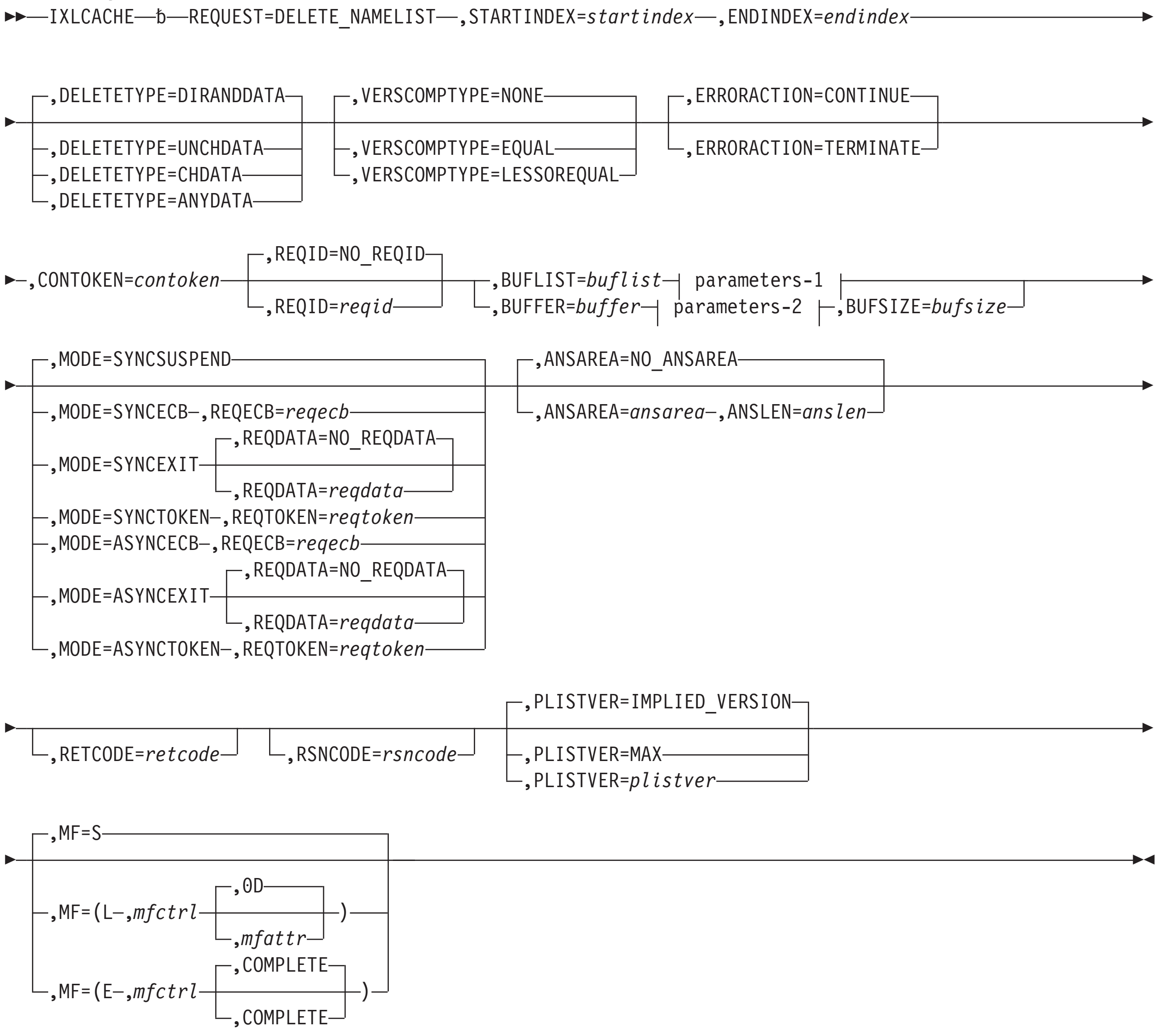
A DELETE_NAMELIST request can be issued only for cache structures allocated in a coupling facility of CFLEVEL=5 or higher. DELETE_NAMELIST requests issued for a cache structure allocated in a coupling facility of CFLEVEL=0 through 4 will fail.

Syntax Diagram

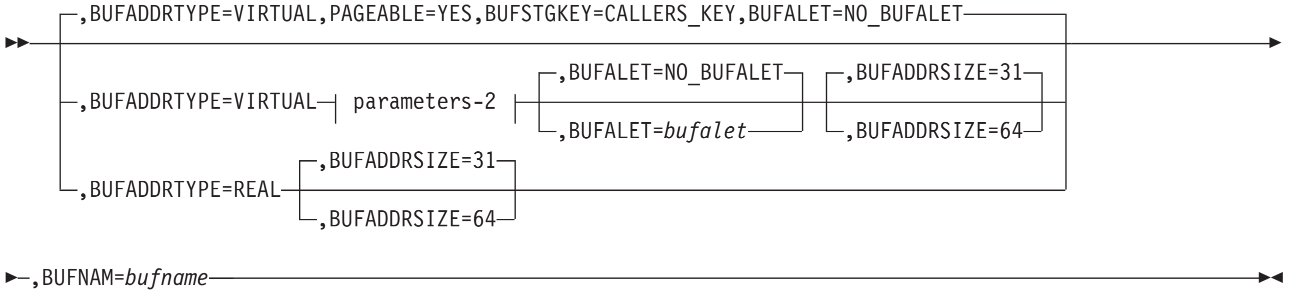
The syntax for the IXLCACHE REQUEST=DELETE_NAMELIST is as follows:

IXLCACHE Macro

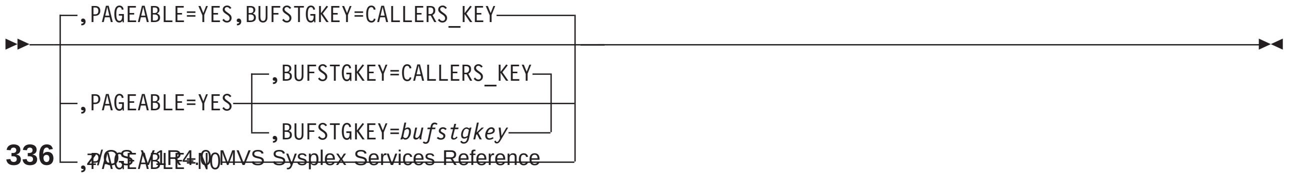
main diagram



parameters-1



parameters-2



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea* and ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=DELETE_NAMELIST are listed in alphabetical order. Default values are underlined:

REQUEST=DELETE_NAMELIST
Use this input parameter to specify that you want to delete a set of entries from the structure. Each entry is represented by a name block, mapped by the IXLYDNNB macro, stored in the area specified by BUFFER or the buffers specified by BUKLIST.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the answer area information returned from the request will be stored.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to contain the returned name blocks tha define the entries that are to be deleted.

The format of the registration block is described by mapping macro IXLYDNNB.

You can define the buffer size to be a total of up to 65536 bytes, but it should not be larger than what you actually require to hold the maximum number of name blocks, subject to the other buffer length requirements stated below.

Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer. See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the name blocks.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold the name blocks identifying the entries to be deleted from the structure. BUFLIST specifies a 128-byte storage area that consists of a list of 1 to 16 buffer address elements.

The format of the name block is described by mapping macro IXLYDNNB.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space as defined by BUFALET.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes as defined by BUFINCRNUM.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. XES treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 1 to 16.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (1 to 16) in the list (BUFLIST)

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer are in the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage key in the format kkkkxxxx, where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

DELETETYPE=DIRANDDATA
DELETETYPE=UNCHNDATA
DELETETYPE=CHDATA
DELETETYPE=ANYDATA
Use this input parameter to specify the type of delete processing that is to be performed.

DIRANDATA
For each applicable structure entry, invalidate the name and remove the name from storage and cast-out classes, and release all directory and data entry resources for the structure for reuse by the structure. The system deregisters interest in all connections with registered interest in the entry and performs a cross-invalidate against their local caches.

UNCHNDATA
For each applicable structure entry, if the data is unchanged, release the data entry resources for the entry for reuse by the structure. The system does not delete the directory entry and does not perform a cross-invalidate.

CHDATA
For each applicable structure, if the data is changed, release the data entry resources for the entry for reuse by the structure, set the change bit, the castout lock, and the castout lock state to zero, and remove the directory entry from the cast-out class. The system does not delete the directory entry and does not perform a cross-invalidate.

ANYDATA
For each applicable structure, whether the data is changed or unchanged, release the data entry resources for the entry for reuse by the structure. If the data is changed, set the change bit, the castout lock, and the castout lock state to zero, and remove the directory entry from the cast-out class. The system does not delete the directory entry and does not perform a cross-invalidate.

,ENDINDEX=*endindex*
Use this input parameter to specify the ending index for name block processing. The index value must be greater than or equal to the value specified for STARTINDEX.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field containing the ending index value.

ERRORACTION=CONTINUE
ERRORACTION=TERMINATE
Use this input parameter to specify whether processing is to continue with the next name block if an entry is not found or a version mismatch occurs.

CONTINUE
If an error occurs, processing is to continue with the next name block.

TERMINATE
If an error occurs, processing is to halt and the index of the name block that caused the error is returned in the answer area.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN

The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB

The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT

The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN

The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES

,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.

- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*
Use this input parameter to specify the starting index for name block processing. Valid STARTINDEX values are from 1 to the value of ENDINDEX. The first registration block has index number 1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field containing the starting index value.

VERSCOMPTYPE=NONE
VERSCOMPTYPE=EQUAL
VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how the structure entry version number comparison is to be performed.

VERSCOMPTYPE=NONE
No version number is provided for comparison, therefore a version number comparison is not to be performed.

VERSCOMPTYPE=EQUAL
The version number for the structure entry must be equal to the value specifed in the corresponding name block mapped by IXLYDNNB.

VERSCOMPTYPE=LESSOREQUAL
The version number for the structure entry must be less than or equal to the value specified in the corresponding name block mapped by IXLYDNNB.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the next registration block to be processed has been returned in the answer area (field CAARNLINDEX). **Action:** Process the entries that have completed on the current request and then reissue the request. The entries that were processed are indexed from STARTINDEX to CAARNLINDEX-1. To restart the request, update STARTINDEX with the value of CAARNLINDEX, the index of the next name element to be processed. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The vector index in the registration block indexed by CAARNLINDEX is not valid. None of the registration blocks have been processed. All vector indexes for all registration blocks in the buffer have been set to indicate that the local buffer is not valid. **Action:** Correct the vector index value and reissue the REG_NAMELIST request with the same STARTINDEX and ENDINDEX values.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. A REG_NAMELIST registration block specified a storage class value that exceeded the maximum defined storage class for the structure. CAARNLINDEX contains the index of the registration block that contained the storage class value in error. All registration blocks preceding this block were processed. **Action:** Correct the storage class value in the registration block indexed by CAARNLINDEX and reissue the REG_NAMELIST request starting with that registration block.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If the caller is running in AR-mode and the BUFFER parameter was specified using explicit register notation, the corresponding access register was updated appropriately.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER is not addressable. All bits in the local cache vector may be reset to overindicate that the data items are not valid. **Action:** Ensure that: • The correct buffer address was used for BUFFER. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • The buffer area is addressable in the caller's primary address space or from the caller's PASN access list. • If the caller is running in AR-mode and the BUFFER parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • Requirements for BUFFER and BUFSIZE. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0874	**Equate Symbol:** IXLRSNCODEBADRNLINDEX **Meaning:** Program error. Either the value specified for either STARTINDEX or ENDINDEX is not valid or the size of the buffer specified by BUFFER is smaller than required based on the value of ENDINDEX. **Action:** Ensure that: • The values specified by STARTINDEX and ENDINDEX are in the range of 1 to 32. • The value specified by ENDINDEX is greater than or equal to the value of STARTINDEX. • The value specified by ENDINDEX does not imply a larger BUFFER size than was actually specified on the REG_NAMELIST request.
8	xxxx0875	**Equate Symbol:** IXLRSNCODEBADNSBAREA **Meaning:** Program error. The storage area specified by NSBAREA is not addressable. **Action:** Ensure that: • The address passed in NSBAREA is valid. • The NSBAREA area was not previously freed. • The NSBAREA area is addressable from the caller's primary address space or from the caller's PASN access list. • If the caller is running in AR-mode and NSBAREA was specified using explicit register notation, ensure that the corresponding access register was updated appropriately. • If the caller is disabled, then NSBAREA must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXLCACHE macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. Allocation of a directory entry was necessary, but was unavailable or could not be reclaimed. In the answer area, CAASTGCLFULL contains the storage class from which the reclaiming operation failed. CAARNLINDEX contains the index of the registration block that was being processed when the error occurred. All prior registration blocks were processed. **Action:** • Determine if any data items may be cast-out to make room for this item. • Check your usage of storage classes to see if some data items can be moved to a different storage class (preferably with a lower priority) so some entries in the structure can be freed. • Determine if a rebuild or an alter of the structure is necessary to make room for more data entries/items. After correcting the error, restart the REG_NAMELIST request starting with the registration block indexed by CAARNLINDEX.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.

Table 26. Return and Reason Codes for the IXLCACHE REQUEST=DELETE_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either continue processing using single READ_DATA requests to perform the function of the REG_NAMELIST request or disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=2 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=PROCESS_REFLIST

Description

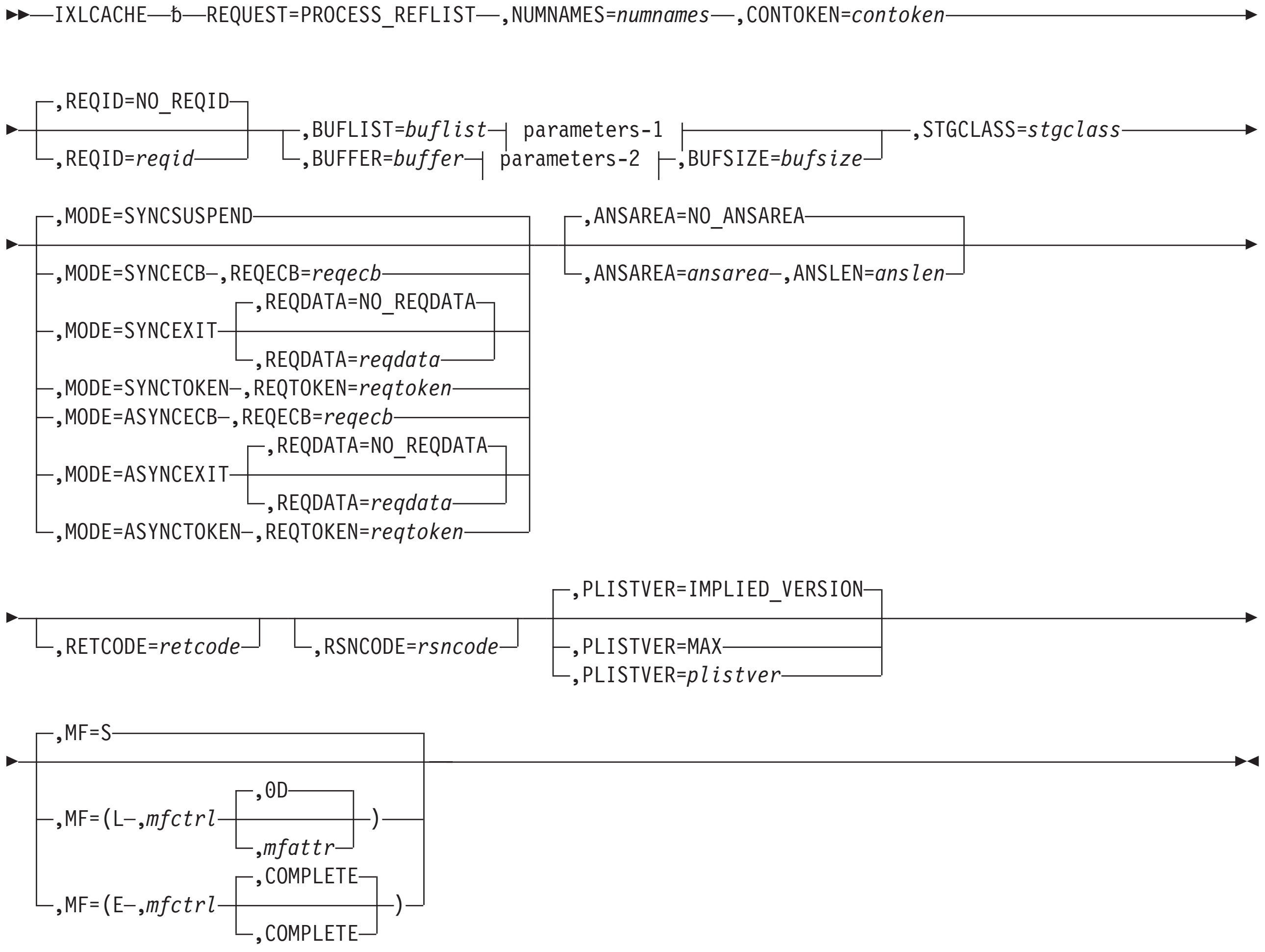
A PROCESS_REFLIST request allows you to indicate that a set of data items has been recently referenced, even in the absence of actual read or write activity against those data items. This causes the storage class queue to be re-ordered and the data items in the list to be considered recently referenced. The least recently referenced data items are the first to be considered for a reclaim. See *z/OS MVS Programming: Sysplex Services Guide* for more information on reclamation and referenced versus unreferenced.

The list of data items to be processed should be supplied in the storage area specified by BUFLIST or BUFFER. You must also specify the number of names (NUMNAMES) contained in the buffer(s). The data items are processed only if the connection has registered interest in the data items and the data items are associated with the specified storage class (STGCLASS).

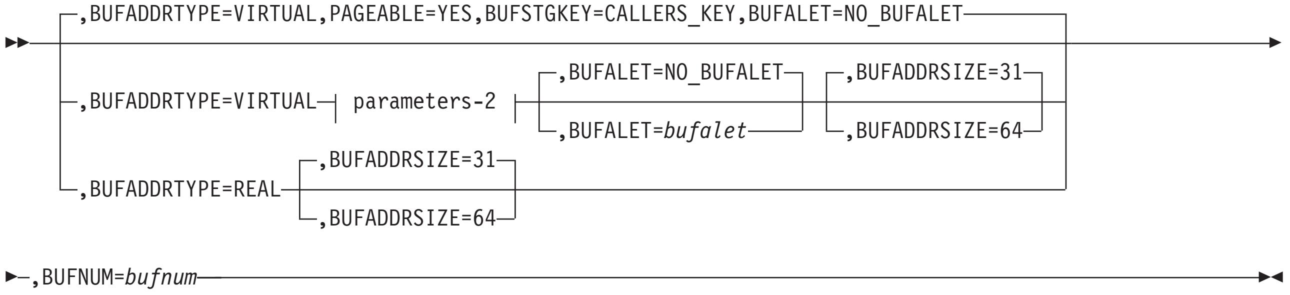
Syntax Diagram

The syntax of the IXLCACHE REQUEST=PROCESS_REFLIST macro is as follows:

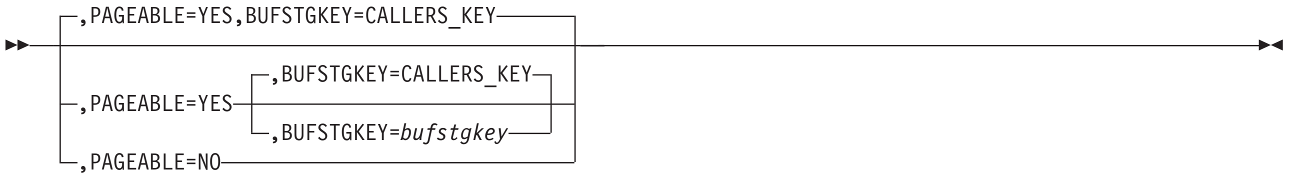
main diagram



parameters-1



parameters-2



Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=PROCESS_REFLIST are listed in alphabetical order. Default values are underlined:

REQUEST=PROCESS_REFLIST
Use this input parameter to specify that the data items listed by name in the storage areas specified by BUFLIST or BUFFER be processed as having been recently referenced.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=<u>31</u>
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to contain a list of entry names to be processed.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.
- Does not start below storage address 512.

The buffers must be formatted to contain 16-byte elements, starting at offset zero. Each 16-byte element must contain a name to be processed.

See the NUMNAMES description to specify the number of data items to be processed, and the BUFSIZE description to specify the size of the buffer.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the list of names to be processed.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold a list of names to be processed. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.
- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.

- Not start below storage address 512.
- Consist of 16-byte elements, starting at offset zero. Each 16-byte element should contain a name to be processed. The named element must reside in the storage class specified by STGCLASS.

Note: The buffers do not have to be contiguous in storage. Cache services treat BUFLIST buffers as a single buffer, even if the buffers are not contiguous.

See the NUMNAMES description to specify the number of data items to be processed, and the BUFNUM parameter to specify the number of buffers in the buffer list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no entries will be processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NUMNAMES=*numnames*
Use this input parameter to specify the number of names contained in the BUFFER area or BUFLIST buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the number of names.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STGCLASS=*stgclass*

Use this input parameter to specify the storage class to which the data items to be processed must be assigned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage class.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified by STGCLASS exceeds the maximum number of storage classes defined for the cache structure. **Action:** The number of storage classes allowed for a structure are defined on the IXLCONN macro by the NUMSTGCLASS parameter. Correct the STGCLASS parameter to specify a valid storage class.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0830	**Equate Symbol:** IXLRSNCODEBADNUMNAMES **Meaning:** The NUMNAMES specification is not valid. **Action:** Ensure that the number of names specified by NUMNAMES reflects the correct number of names stored in BUFLIST or BUFFER.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the caller's PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information).
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.

Table 27. Return and Reason Codes for the IXLCACHE REQUEST=PROCESS_REFLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=READ_COCLASS

Description

A READ_COCLASS request allows you to read directory entry names and directory entry user data entries associated with a specified cast-out class (COCLASS). You may filter the cast-out class information by specifying the NAME and NAMEMASK parameters. You may limit the amount of information returned with the DIRINFOFMT parameter. The retrieved information is stored in the storage areas specified by BUFLIST or BUFFER. For structures allocated in a coupling facility of CFLEVEL=4 or lower, the format of the returned information is described by mapping macro IXLYCANB.

For structures allocated in a coupling facility of CFLEVEL=5 or higher, the format of the returned information is described by either mapping macro IXLYCANB or mapping macro IXLYDEIB, depending on the value specified for the DIRINFOFMT keyword.

The number of directory entry name elements (names and user data) that are retrieved is provided in the answer area (ANSAREA). The answer area is described by the IXLYCAA mapping macro.

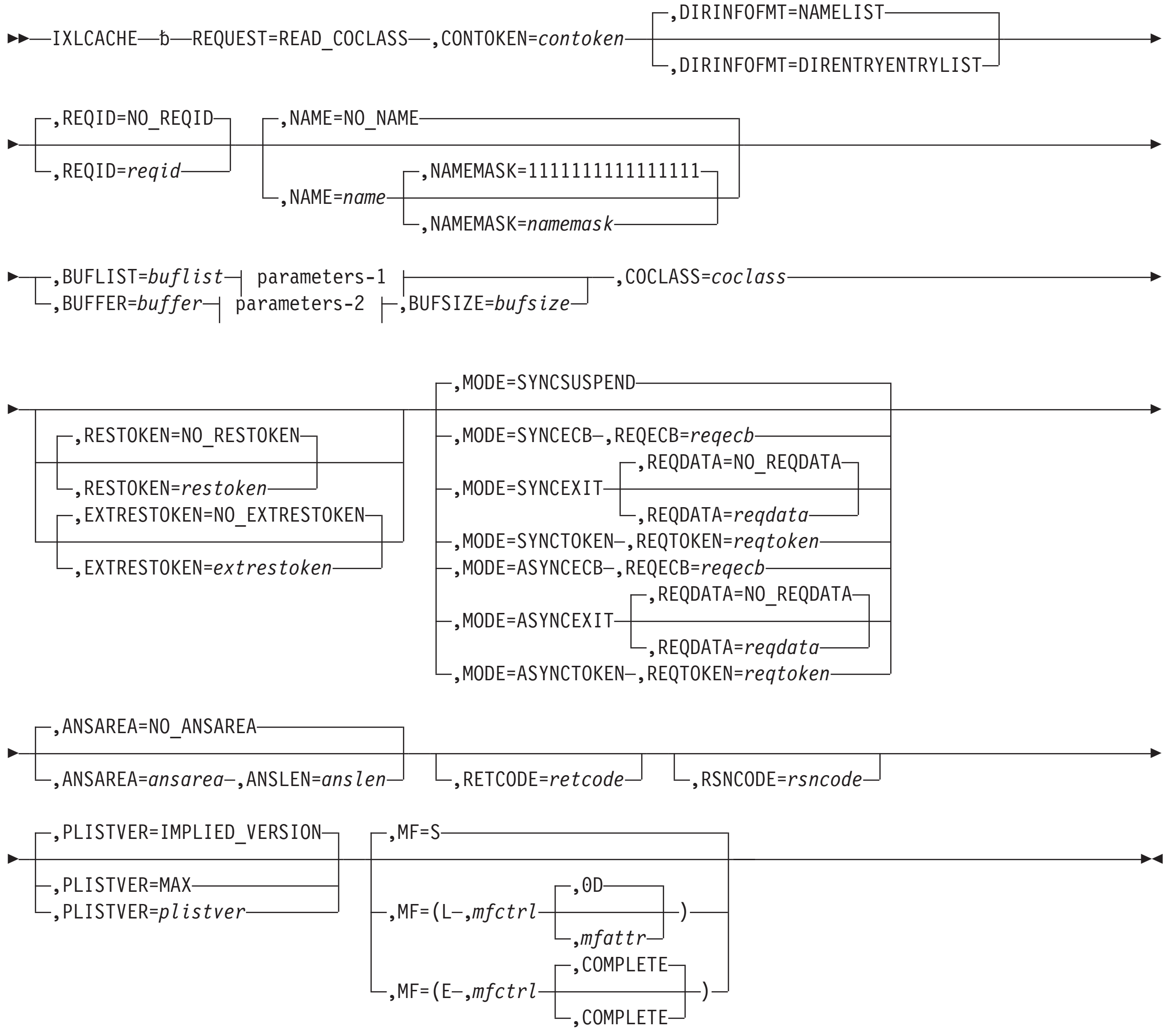
If the request completes prematurely because it exceeds the model-dependent time-out criteria or the specified buffer area (BUFLIST or BUFFER) is full, a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area (fields CAARESTOKEN or CAAEXTRESTOKEN). The token can be specified on the next READ_COCLASS request to resume processing with the next directory entry to be processed. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

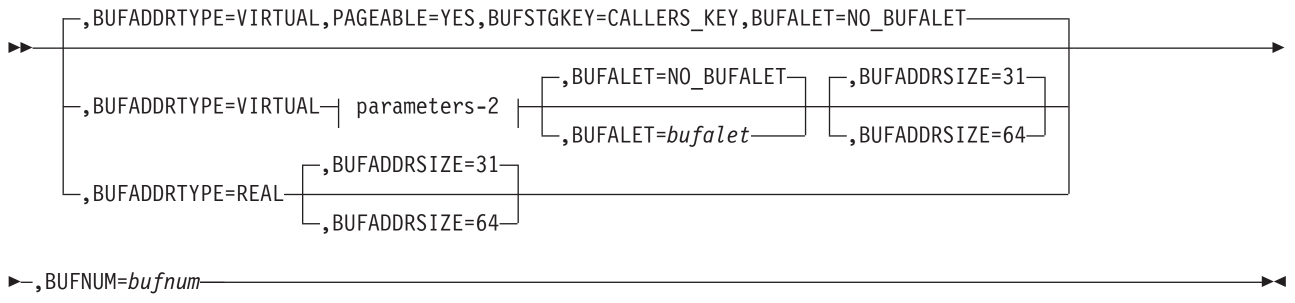
The syntax diagram for IXLCACHE REQUEST=READ_COCLASS is as follows:

IXLCACHE Macro

main diagram



parameters-1



parameters-2

```
   ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────┐
►►─┼──────────────────────────────────────────┼──────────────────────────────────────────►◄
   │                ┌─,BUFSTGKEY=CALLERS_KEY─┐│
   ├─,PAGEABLE=YES──┼────────────────────────┼┤
   │                └─,BUFSTGKEY=bufstgkey───┘│
   └─,PAGEABLE=NO─────────────────────────────┘
```

Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea*,ANSLEN=*anslen* is required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_COCLASS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_COCLASS
Use this input parameter to specify that the directory entry names and information for a specified cast-out class (COCLASS) be stored in the areas specified by BUFFER or BUFLIST. You may filter the amount of information returned by the request by using the NAME and NAMEMASK parameters.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The information can include a restart token or extended restart token when the request exceeds the model-dependent time-out criteria or the specified buffer area is full. The format of the answer area is described by the IXLYCAA mapping macro.

Upon successful completion of the request the answer area contains the number of directory entries processed by the request (field CAADIRCOUNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this output parameter to specify a buffer area to contain the data that is read from the specified cast-out class.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.
- Does not start below storage address 512.

Use the BUFSIZE description for specifying the size of the buffer.

Upon completion of the request, the buffer is arranged in 32-byte segments, starting at offset zero. The mapping of the buffers is described by the IXLYCANB mapping macro. For each named data item reported on, the following information is returned to the buffer:
- The data item name
- The user data in the directory entry
- The size of the data entry (that is, the number of elements in the data entry)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to receive the requested information.

,BUFLIST=*buflist*
Use this output parameter to specify a list of buffers to hold the data that is read from the specified cast-out class. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.
- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. Cache services treat BUFLIST buffers as a single buffer even if the buffers are not contiguous.

Use the BUFNUM parameter to specify the number of buffers in the buffer list.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

Upon completion of the request, the buffers are arranged in 32-byte segments, starting at offset zero. The mapping of the buffers is described by the IXLYCANB mapping macro. For each named data item reported on, the following information is returned to the buffers:
- The data item name
- The user data in the directory entry
- The size of the data entry

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=<u>CALLERS_KEY</u>
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,COCLASS=*coclass*

Use this input parameter to specify the cast-out class for which directory names and directory entry user data should be retrieved.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the cast-out class value.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DIRINFOFMT=<u>NAMELIST</u>
,DIRINFOFMT=DIRENTRYLIST

Use this input parameter to specify the amount of information to be returned for each directory entry.

DIRINFOFMT is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

NAMELIST
Return a subset of the information for each directory. The information returned includes:
- Name
- User data
- Size of the data entry.

DIRENTRYLIST
Return all information for each directory. The information returned includes:
- Name
- User data
- Storage class assigned to
- Indication of whether data is currently cached
- Indication of whether data is changed (if cached)
- Castout class assigned to (if changed)
- Parity bits
- Value and state of the cast-out lock
- Indication of which connected users have registered interest
- Size of the data entry
- Version number.

See the IXLYDEIB macro for the format of the data returned in the BUFFER area or the BUFLIST buffers.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*
Use this input parameter to specify an extended restart token that can be used to resume processing of a READ_COCLASS request that completed prematurely because it exceeded the model-dependent time-out criteria or because the specified buffer area is full. The extended restart token is returned in the answer area (field CAAEXTRESTOKEN), and should be specified on the next READ_COCLASS request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria or the buffer does not become full, the extended restart token will not be provided.

Notes:
1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.
3. Specifying an extended restart token requires that the length of the answer area be at least the length of CAALEVEL1LEN.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an

asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=NO_NAME
,NAME=*name*
Use this input parameter to filter by name, the data item for which directory entry data will be read. You may use this parameter along with the NAMEMASK parameter to select certain data items. See the NAMEMASK parameter for more information on this option.

The entry name and user data associated with NAME, possibly NAMEMASK, and the specified cast-out class (COCLASS) are retrieved. If NAME is not specified, all the data items in the specified cast-out class are read.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NAMEMASK=1111111111111111
,NAMEMASK=*namemask*
Use this input parameter to specify which characters in the name specified by NAME are to be used in selecting data items for processing. This parameter allows you to select multiple data items based on common characters in NAME.

The position of each bit in NAMEMASK corresponds to the same relative character position in NAME. A one indicates that the corresponding letter should be used in selecting entries; a zero indicates that the corresponding letter should not be used.

Specifying a name mask with all zeros causes all names to be selected for processing. Specifying a name mask with all ones causes only the name specified by NAME to be selected.

For more information on how NAMEMASK may be used, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the bit-mask for the name specified on the NAME keyword.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*

Use this input parameter to specify a restart token that can be used to resume processing of a READ_COCLASS request that completes prematurely because it exceeds the model-dependent time-out criteria or the specified buffer area (BUFLIST or BUFFER) is full. The restart token is returned in the answer area, and should be specified on the next READ_COCLASS request to resume processing with the next directory entry to be processed.

If the request does not exceed the model-dependent time-out criteria or the buffer does not become full, the returned token will contain all binary zeros.

Notes:

1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify a restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The following information has been returned in the answer area: • The number of elements returned in the BUFFER or BUFLIST area (field CAADIRCOUNT) • A token for restarting the request (field CAARESTOKEN or CAAEXTRESTOKEN). **Action:** Reissue the request using the restart token (RESTOKEN or CAAEXTRESTOKEN). For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** The request completed prematurely because the buffer specified by BUFFER, or the buffers specified by BUFLIST, is full. The following information has been returned in the answer area: • The number of elements returned in the BUFFER or BUFLIST area (field CAADIRCOUNT) • A token for restarting the request (field CAARESTOKEN or CAAEXTRESTOKEN) **Action:** Reissue the request, or increase the size of the buffer(s) and rerun your program. You can reissue the request using the restart token (RESTOKEN or EXTRESTOKEN). **Note:** Process the data that has been returned in the buffers prior to reissuing the request. The data returned from the original request will be overwritten if you specify the same buffer address. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082E	**Equate Symbol:** IXLRSNCODEBADCOCLASS **Meaning:** Program error. The cast-out class specified by COCLASS exceeds the maximum number of cast-out classes defined for the cache structure. **Action:** Correct the COCLASS parameter to specify a valid cast-out class. The valid range for COCLASS is from 1 to the maximum value specified by the NUMCOCLASS parameter on the IXLCONN macro.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to resume the request. Possible causes are: • The entry to be resumed on no longer resides in the cast-out class specified by COCLASS. • The specified token does not correspond to the restart token returned in the answer area of the previous request. • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size and reissue the request. Consider using the BUFLIST parameter if BUFFER was specified.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. **Action:**Discard the results of the initial request and reissue the request with an EXTRESTOKEN value of zero. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 28. Return and Reason Codes for the IXLCACHE REQUEST=READ_COCLASS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=READ_COSTATS

Description

A READ_COSTATS request allows you to retrieve statistical information for the specified cast-out classes in the range beginning with COCLASSB and ending with COCLASSE. The information is returned in the storage areas specified by BUFLIST or BUFFER. The format of the information returned depends on the level of the coupling facility in which the structure is allocated and on the COSTATSFMT specification. For cache structures allocated in a coupling facility of CFLEVEL=5 or higher, you can use the COSTATSFMT keyword to specify the level of detailed information that is to be returned.

For structures allocated in a coupling facility of CFLEVEL=4 or lower, or if COSTATSFMT=COCOUNTSLIST is specified, the format of the information returned is described by the CCIH and CCIHCOUNTS mappings of the IXLYCCIH mapping macro and consists of:
- The first and last cast-out classes for which information was returned.
- For each cast-out class for which information was returned, the number of data elements associated with data entries in the indicated cast-out class.

For structures allocated in a coupling facility of CFLEVEL=5 or higher, when COSTATSFMT=COSTATSLIST, the format of the information returned is described by the CCIH, CCIHCOSTATSLIST, and CCIHCCIBS mappings of the IXLYCCIH mapping macro. The information returned consists of:
- The first and last cast-out classes for which information was returned.
- For each cast-out class for which information was returned, the number of data elements associated with data entries in the indicated cast-out class and the user data that was written to the directory entry when the IXLCACHE WRITE_DATA request was invoked. If the structure was allocated with a UDF order queue, the field contains the user data of the first entry on the UDF order queue. If the structure was allocated without UDF order queues, the field contains the user date of the first entry on the cast-out class queue.

  (The UDFORDER keyword of IXLCONN is used to allocate a cache structure with UDF order queues.)

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=READ_COSTATS is as follows:

main diagram

```
►►──IXLCACHE──␢──REQUEST=READ_COSTATS──,COCLASSB=coclassb──,COCLASSE=coclasse──────────────────────►


   ┌─,COSTATSFMT=COCOUNTSLIST─┐                          ┌─,REQID=NO_REQID─┐
►──┼──────────────────────────┼──,CONTOKEN=contoken──────┼─────────────────┼────────────────────────►
   └─,COSTATSFMT=COSTATSLIST──┘                          └─,REQID=reqid────┘

                                                                ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►──┬─,BUFLIST=buflist─┤ parameters-1 ├──────────────────────────┼────────────────────────────────────────────┼──►
   └─,BUFFER=buffer─┤ parameters-2 ├─,BUFSIZE=bufsize─┘         ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
                                                                │               ┌─,REQDATA=NO_REQDATA─┐      │
                                                                ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
                                                                │               └─,REQDATA=reqdata────┘      │
                                                                ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
                                                                ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
                                                                │                ┌─,REQDATA=NO_REQDATA─┐     │
                                                                ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
                                                                │                └─,REQDATA=reqdata────┘     │
                                                                └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┘

   ┌─,ANSAREA=NO_ANSAREA──────────────────┐
►──┼──────────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬──────────────────────►
   └─,ANSAREA=ansarea─,ANSLEN=anslen──────┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────────┼──────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                      ┌─,0D──────┐          │
   └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl───────┼──────────┼─)───────┤
                                  │                      └─,mfattr──┘          │
                                  │                      ┌─,COMPLETE─┐         │
                                  └─,MF=(E─,mfctrl───────┼───────────┼─)───────┘
                                                         └─,COMPLETE─┘
```

parameters-1

```
     ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─────────────────┐
►►───┼────────────────────────────────────────────────────────────────────────────────────────────┼──────►
     │                                                 ┌─,BUFALET=NO_BUFALET─┐ ┌─,BUFADDRSIZE=31─┐ │
     ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├───────────┼─────────────────────┼─┼─────────────────┼─┤
     │                                                 └─,BUFALET=bufalet────┘ └─,BUFADDRSIZE=64─┘ │
     │                        ┌─,BUFADDRSIZE=31─┐                                                  │
     └─,BUFADDRTYPE=REAL──────┼─────────────────┼──────────────────────────────────────────────────┘
                              └─,BUFADDRSIZE=64─┘

►─,BUFNUM=bufnum──────────────────────────────────────────────────────────────────────────────────────►◄
```

parameters-2

```
     ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────┐
►►───┼──────────────────────────────────────────┼─────────────────────────────────────────────────────►◄
     │                 ┌─,BUFSTGKEY=CALLERS_KEY─┐│
     ├─,PAGEABLE=YES───┼────────────────────────┼┤
     │                 └─,BUFSTGKEY=bufstgkey───┘│
     └─,PAGEABLE=NO─────────────────────────────┘
```

Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_COSTATS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_COSTATS
Use this input parameter to specify that statistical information for the cast-out classes in the range COCLASSB to COCLASSE be retrieved and placed in the storage areas specified by BUFLIST or BUFFER.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=<u>31</u>
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=<u>VIRTUAL</u>
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=<u>NO_BUFALET</u>
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the returned cast-out class statistics.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.
- Does not start below storage address 512.

  See the BUFSIZE description for specifying the size of the buffer.

Upon a successful completion of the request:
- The first four bytes of the buffer contains the start and end cast-out classes for which information was returned. (The format of these four bytes is described by mapping macro IXLYCCIH.)
- Beginning at offset four, the buffer contains an array of fullwords, each corresponding to a cast-out class in the specified range (CCIHCOCLASSBEG to CCIHCOCLASSEND). (Each fullword corresponds to a cast-out class by relative position, such that, for example, the first fullword in the array corresponds to the first cast-out class in the range, and so forth.) Each fullword contains the number of data elements associated with the data entries assigned to this cast-out class.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) (the buffer) to contain the data returned from the request.

,BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold the returned cast-out class statistics. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.
- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. Cache services treat BUFLIST buffers as a single buffer, even if the buffers are not contiguous.

See the BUFNUM parameter to specify the number of buffers in the buffer list.

Upon successful completion of the request:
- The first four bytes of the buffers contain the start and end cast-out classes for which information was returned. (The format of these four bytes is described by mapping macro IXLYCCIH.)
- Beginning at offset four in the first buffer, the buffers contain an array of fullwords, each corresponding to a cast-out class in the specified range (CCIHCOCLASSBEG to CCIHCOCLASSEND). (Each fullword corresponds to a cast-out class by relative position, such that, for example, the first fullword in the array corresponds to the first cast-out class in the range, and so forth.) Each fullword contains the number of data elements associated with the data entries assigned to this cast-out class.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read from the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller’s PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,COCLASSB=*coclassb*
Use this input parameter to specify the first cast-out class in the range of cast-out classes for which statistics will be reported. The specified value must fall within the range 1 to the maximum number of cast-out classes defined for the structure. The value cannot be larger than that specified for COCLASSE.

Note: The maximum number of cast-out classes is specified on the IXLCONN macro by the NUMCOCLASS parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the first cast-out class.

,COCLASSE=*coclasse*
Use this input parameter to specify the last cast-out class in the range of cast-out classes for which statistics will be reported. The specified value must fall within the range 1 to the maximum number of cast-out classes defined for the structure. The value cannot be smaller than that specified for COCLASSB.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the last cast-out class.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,COSTATSFMT=COCOUNTSLIST
,COSTATSFMT=COSTATSLIST
Use this input parameter to specify the format of the data that is returned for each cast-out class.

COSTATSFMT is meaningful only for structures allocated in a coupling facility with CFLEVEL=5 or higher.

COCOUNTSLIST
Return the data in the BUFFER area or the BUFLIST buffers in the format defined by the CCIH and CCIHCOUNTS mappings of the IXLYCCIH macro.

COSTATSLIST
Return the data in the BUFFER area or the BUFLIST buffers in the format defined by the CCIH, CCIHCOSTATSLIST, and CCIHCCIBS mappings of the IXLYCCIH macro.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=<u>IMPLIED_VERSION</u>
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=<u>NO_REQDATA</u>
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=<u>NO_REQID</u>
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx040B	**Equate Symbol:** IXLRSNCODEHIGHCOEND **Meaning:** The specified COCLASSE exceeds the maximum defined cast-out class for the structure. The range of classes from that specified by COCLASSB through the maximum defined class was reported. **Action:** None. However, if you did not intend for the range to end with the maximum defined cast-out class, specify a value for COCLASSE that is within the maximum limit. The number of cast-out classes is defined on the IXLCONN macro with the keyword NUMCOCLASS.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** The request completed prematurely because the buffer specified by BUFFER, or the buffers specified by BUFLIST, is full. A subrange of the requested cast-out classes was reported on, as described by the IXLYCCIH mapping macro. **Action:** Reissue the request specifying the same value for the end cast-out class (COCLASSE), but a start cast-out class (COCLASSB) equal to the end cast-out class plus one (CCIHCOCLASSEND+1). **Note:** Process the data that has been returned in the buffers prior to reissuing the request. The data returned from the original request will be overwritten if you specify the same buffer address. When you have processed the data returned from the original request, you should update COCLASSB to point to the first unprocessed cast-out class. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082A	**Equate Symbol:** IXLRSNCODEBADCOBEG **Meaning:** Program error. The specified COCLASSB exceeds the maximum number of cast-out classes defined for the structure. **Action:** Specify for COCLASSB a valid index within the range 1 to the maximum cast-out classes defined for the structure. The maximum number of cast-out classes is defined on the IXLCONN macro by the NUMCOCLASS keyword.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size and reissue the request. Consider using the BUFLIST parameter if BUFFER was specified.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.

Table 29. Return and Reason Codes for the IXLCACHE REQUEST=READ_COSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=READ_DATA

Description

A READ_DATA request allows you to read a data item from a data entry in the cache structure and optionally place it in the storage areas specified by BUFFER or BUFLIST and/or ADJAREA.

A data item you attempt to read might not reside in the cache structure at the time the read is requested. (For instance, only a directory entry exists, or neither a directory entry nor a data entry exists.) If this is the case, no data is read. However, you can use a READ_DATA request to allocate a directory entry to the data item. To do this, use the ASSIGN=YES option. This is the method directory-only users employ to allocate a coupling facility cache structure directory entry for a data item.

The NAME parameter identifies the data item that will be, or has already been, introduced to the structure. Once introduced to the structure, NAME also references the structure resources (the directory entry and data entry) allocated for that data item.

With a structure allocated in a coupling facility of CFLEVEL=5 or higher, when issuing a READ_DATA request you can choose whether to register interest in the data item. The default, if you do not specify an option, is to register interest in the data item.

- You dan register or update your interest in the data item by specifying REGUSER=YES and then specifying an index into your local cache vector (VECTORINDEX) to be associated with the data item in the structure. This index identifies a vector entry that cache services uses to indicate both your interest in the data item and the validity of your locally cached copy of the data item. Having registered interest in a data item **means** that you have a valid copy of the data item in your local cache buffer. When another user updates the data item in the structure, cache services will update the associated vector entry, thereby deregistering your interest in the data item and invalidating your locally cached copy.

  Cache services is responsible for deregistering your interest in a data item whenever another user updates the data item in the structure. You are responsible for maintaining the association between the entry in your local cache vector and the copy of the data item in your local cache buffer.
- You can specify that registration of your interest in the data item is not to be performed (REGUSER=NO). Be aware that entries in a data item for which no interest has been registered are higher-priority candidates for reclaim processing.

For cache structures allocated in a coupling facility with CFLEVEL=4 or higher, you can specify RETURNDATA=NO to suppress the read function when a REQUEST=READ_DATA is issued. By specifying RETURNDATA=NO, the READ_DATA request will register interest in the entry without returning the associated data. This is useful when data is cached for the entry but you do not want the data to be read. Note that if adjunct is supported by the structure (ADJUNCT=YES on the IXLCONN invocation), data is cached for the entry, and you have specified ADJAREA, then the adjunct will be returned. The CAAADJAREAVALID bit in the cache answer area is set to indicate whether adjunct data was returned.

Syntax Diagram

The syntax for the IXLCACHE REQUEST=READ_DATA is as follows:

IXLCACHE Macro

main diagram

```
                                     ┌─,ASSIGN=YES─┐
►►──IXLCACHE──ƀ──REQUEST=READ_DATA───┼─────────────┼──────────────────────────────────────────────►
                                     └─,ASSIGN=NO──┘


                     ┌─,OLDNAME=NO_OLDNAME──────────────────────────┐
  ┌─,REGUSER=NO──────┼──────────────────────────────────────────────┼──────────┐
  │                  └─,OLDNAME=oldname─,VECTORINDEX=vectorindex────┘          │  ┌─,RETURNDATA=YES─┐
►─┼────────────────────────────────────────────────────────────────────────────┼──┼─────────────────┼──►
  │                   ┌─,OLDNAME=NO_OLDNAME─┐                                  │  └─,RETURNDATA=NO──┘
  └─,REGUSER=YES──────┼─────────────────────┼─,VECTORINDEX=vectorindex─────────┘
                      └─,OLDNAME=oldname────┘

                          ┌─,REQID=NO_REQID─┐                ┌─,OLDNAME=NO_OLDNAME─┐
►─,CONTOKEN=contoken──────┼─────────────────┼──,NAME=name────┼─────────────────────┼──────────────►
                          └─,REQID=reqid────┘                └─,OLDNAME=oldname────┘


                                                        ┌─,ADJAREA=NO_ADJAREA─┐
►─┬──────────────────────────────────────────────────┬──┼─────────────────────┼─,STGCLASS=stgclass──►
  ├─,BUFLIST=buflist─┤ parameters-1 ├─┬──────────────┤  └─,ADJAREA=adjarea────┘
  └─,BUFFER=buffer─┤ parameters-2 ├───┴─,BUFSIZE=bufsize─┘


                              ┌─,MODE=SYNCSUSPEND──────────────────────────────┐
►─,VECTORINDEX=vectorindex────┼────────────────────────────────────────────────┼──────────────────────►
                              ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤
                              │               ┌─,REQDATA=NO_REQDATA─┐          │
                              ├─,MODE=SYNCEXIT┼─────────────────────┼──────────┤
                              │               └─,REQDATA=reqdata────┘          │
                              ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
                              ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
                              │                ┌─,REQDATA=NO_REQDATA─┐         │
                              ├─,MODE=ASYNCEXIT┼─────────────────────┼─────────┤
                              │                └─,REQDATA=reqdata────┘         │
                              └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┘


  ┌─,ANSAREA=NO_ANSAREA──────────────────┐
►─┼──────────────────────────────────────┼──┬───────────────────┬──┬───────────────────┬──────────────►
  └─,ANSAREA=ansarea─,ANSLEN=anslen──────┘  └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘


  ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S──────────────────────────────────┐
►─┼───────────────────────────┼───┼────────────────────────────────────────┼──────────────────────►◄
  ├─,PLISTVER=MAX─────────────┤   │                    ┌─,0D──────┐        │
  └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl─────┼──────────┼─)──────┤
                                  │                    └─,mfattr──┘        │
                                  │                    ┌─,COMPLETE─┐       │
                                  └─,MF=(E─,mfctrl─────┼───────────┼─)─────┘
                                                       └─,COMPLETE─┘
```

parameters-1

```
   ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET────────┐
►►─┼──────────────────────────────────────────────────────────────────────────────────┼──────────►
   │                                           ┌─,BUFALET=NO_BUFALET─┐ ┌─,BUFADDRSIZE=31─┐│
   ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├──────┼─────────────────────┼─┼─────────────────┼┤
   │                                           └─,BUFALET=bufalet────┘ └─,BUFADDRSIZE=64─┘│
   │                     ┌─,BUFADDRSIZE=31─┐                                              │
   └─,BUFADDRTYPE=REAL───┼─────────────────┼──────────────────────────────────────────────┘
                         └─,BUFADDRSIZE=64─┘

►─,BUFNUM=bufnum──,BUFINCRNUM=bufincrnum─────────────────────────────────────────────────────►◄
```

parameters-2

```
   ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY───────┐
►►─┼───────────────────────────────────────────┼──────────────────────────────────────────►◄
   │                 ┌─,BUFSTGKEY=CALLERS_KEY─┐│
   ├─,PAGEABLE=YES───┼────────────────────────┼┤
   │                 └─,BUFSTGKEY=bufstgkey───┘│
   └─,PAGEABLE=NO──────────────────────────────┘
```

Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_DATA are listed in alphabetical order. Default values are underlined:

REQUEST=READ_DATA

Use this input parameter to specify that the data item identified by NAME be read from the cache structure and stored in the areas specified by BUFFER, BUFLIST and/or ADJAREA.

Directory-only users can also use this parameter to define a new data item to the cache structure and register interest in that data item.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*

Use this output parameter to specify a storage area to contain the adjunct data that was read from the designated entry.

If the structure supports adjunct data and ADJAREA is not specified, or if ADJAREA=NO_ADJAREA is specified, then no adjunct data is returned.

If the structure does not support adjunct data and ADJAREA is specified, then no adjunct data is returned, and the request will complete with a return code X'4' and a reason code of IXLRSNCODENOADJUNCTDATA.

| If the structure supports adjunct data and ADJAREA is specified, but there is no data and adjunct associated with the entry, then:

- For READ_DATA requests, the request completes with return code X'4' and reason code IXLRSNCODENOREADDATA.
- For CASTOUT_DATA requests, the request completes with return code X'8' and reason code IXLRSNCODECOUNCHANGED.

(Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte virtual storage area that the adjunct data will be read into.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro. On a successful completion of the request, the following information is returned in the answer area:

- An indication of whether the entry has changed in the cache structure (field CAACHANGED).
- The parity of the returned data item (field CAAPARITY).
- The cast-out lock state (field CAACOLOCKSTATE).
- The cast-out lock value (field CAACOLOCKVAL).
- If the request replaced a previously specified local cache vector index for the entry, the replaced index is returned (fields CAAINVLCVI and CAAINVLCVINUM).
- If BUFFER or BUFLIST is specified, the number of elements in the retrieved entry is returned (field CAAELEMNUM).
- An adjunct area validity indicator (field CAAADJAREAVALID).

The following additional fields are returned in the answer area when the structure is allocated in a coupling facility with CFLEVEL=4 or higher:

- A data-cached indicator (field CAADATACACHED)
- If BUFFER or BUFLIST is not specified, the number of elements in the entry whether or not BUFFER or BUFLIST was specified (field CAAELEMNUM).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the answer area information returned from the request will be stored.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,ASSIGN=YES
,ASSIGN=NO

Use this input parameter to specify whether a directory entry in the cache structure will be allocated for (assigned to) the data item specified by NAME.

YES

- If a directory entry is not already assigned, this option requests that one be assigned for data item NAME. No data is read.
- If a directory entry is already assigned, the assignment remains. If there is a data entry, its contents are read.

NO

This option assumes a directory entry is already assigned to the data item.

- If a directory entry is not already assigned, a return code X'8', reason code IXLRSNCODENOENTRY is returned.
- If a directory entry is already assigned, the assignment remains. If there is a data entry, its contents are read.

Regardless of the option you choose, if a directory entry is assigned but there is no associated entry data, a return code IXLRETCODEWARNING, reason code IXLRSNCODENOREADDATA is returned.

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the entry data that is read from the cache structure.

You can define the buffer size to be a total of up to 65536 bytes. Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

See the BUFSIZE parameter description for defining the size of the buffer. See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to receive the entry requested from the cache structure.

,BUFINCRNUM=*bufincrnum*
Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*
Use this output parameter to specify a list of buffers to hold the entry data that is read from the cache structure. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer address elements.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space as defined by BUFALET.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes as defined by BUFINCRNUM.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. XES treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

See the BUFALET keyword for information on virtual buffers.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using register 2 to 12) of a 128-byte field that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT

,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN

The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB

The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT

The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN

The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=*name*

Use this input parameter to specify the name of the data item to be read from the cache structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,OLDNAME=NO_OLDNAME
,OLDNAME=*oldname*

Use this input parameter to specify the name of the data item for which your interest should be deregistered.

- For structures allocated in a coupling facility of CFLEVEL=4 or lower, you must also specify the name of the data item for which interest is to be registered after the interest is deregistered in OLDNAME. Identify the data item to which interest is to be registered with NAME. The VECTORINDEX currently associated with the entry specified by OLDNAME will be reassigned to the name of the data item specified by NAME.
- For structures allocated in a coupling facility of CFLEVEL=5 or higher, you can specify that interest in the name of the data item specified by OLDNAME is to be deregistered without registering interest in the entry specified by NAME.

In either case, whenever deregistration of interest is requested, VECTORINDEX must be specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the old data item that was associated with VECTORINDEX.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REGUSER=YES
,REGUSER=NO

Use this input paraameter to specify whether the request should register (or update) interest in the data item to be cast-out.

YES

Connection interest in the data item will be registered for the connection specified by CONTOKEN. If interest is already registered, the vector index specified by VECTORINDEX replaces any previously specified index for the data item.

Specify OLDNAME to deregister any outstanding interest for the specified vector index for a different entry.

NO

Connection interest in the data item will not be registered.

For structures allocated in a coupling facility with CFLEVEL=5 or higher, OLDNAME may be specified to deregister any outstanding interest for the specified local cache vector index for a different entry. If OLDNAME is specified, then VECTORINDEX is required.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RETURNDATA=YES

,RETURNDATA=NO
Use this input parameter to specify whether the data, if any exists for the data entry specified by NAME, is to be read. This option is valid only for cache structures allocated in a coupling facility with CFLEVEL=4 or higher. If ADJAREA is specified and adjunct data is supported by the structure, then the CAAADJAREAVALID bit is set in the answer area:

- CAAADJAREAVALID=B'1' indicates that the system returned adjunct data in ADJAREA.
- CAAADJAREAVALID=B'0' indicates that data was not cached for the data entry specified by NAME or the structure did not support adjunct data.

RETURNDATA=YES
Specifies that the data is to be read into the storage area specified by either BUFFER or BUFLIST and/or ADJAREA.

RETURNDATA=NO
Specifies that for structures allocated in a coupling facility with CFLEVEL=4 or higher, the data entry read function will be suppressed. Adjunct data, if it exists and ADJAREA is specified, will be returned.

If RETURNDATA=NO is specified for a structure allocated in a coupling facility with CFLEVEL=3 or lower, this parameter is ignored and the data, if any exists for the specified NAME, is returned.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STGCLASS=*stgclass*
Use this input parameter to specify a storage class that the entry to be read is to be assigned to. Any previous assignment is updated to the new specification.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage class.

,VECTORINDEX=*vectorindex*
Use this input parameter to specify an index into your local cache vector to be associated with the data item specified by NAME, to be disassociated with the data item specified by OLDNAME, or both. The vector index identifies a vector entry that cache services will use to indicate both your interest in the data item and the validity of the copy of the data item in your local cache buffer.

If the vector index you specify is already associated with another data item, you must disassociate the vector index from the old name by specifying OLDNAME before the vector index can be associated with the data item specified by NAME.

VECTORINDEX is required when REGUSER=YES is specified or defaulted, or whenever OLDNAME is specified. VECTORINDEX is not required when REGUSER=NO is specified without OLDNAME.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the vector index for NAME or OLDNAME.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx040A	**Equate Symbol:** IXLRSNCODENOREADDATA **Meaning:** The request registered interest in the named data item, but either there is no entry data in the cache structure or the read function was suppressed on the request. Therefore, no data is returned in BUFLIST or BUFFER. For structures allocated in a coupling facility with CFLEVEL=4 or higher, if the structure entry contained data and adjunct and ADJAREA was specified, then the adjunct is returned. The CAAADJAREAVALID bit in the answer area is set to B'1' if the adjunct is returned or B'0' if adjunct does not exist for the entry. The following answer area fields are filled in: • CAAADJAREAVALID • CAAINVLCVI • CAAINVLCVINUM For CFLEVEL=4 and higher coupling facilities, the following additional answer area fields are returned: • CAACHANGED • CAACOLOCKSTATE • CAACOLOCKVAL • CAADATACACHED • CAAELEMNUM • CAAPARITY For CFLEVEL=5 and higher coupling facilities, the following additional answer area field is returned: • CAAVERSION **Action:** No action is necessary. If this return code and reason code are unexpected, you may want to: • Verify the NAME of the data item. • Verify that the CONTOKEN is for the correct structure.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040C	**Equate Symbol:** IXLRSNCODENOADJUNCTDATA **Meaning:** The request specified that adjunct data was to be read, but the structure does not support adjunct areas. No adjunct data was retrieved; however, entry data was returned in BUFLIST or BUFFER if requested. The following fields are returned in the answer area: • CAACHANGED • CAACOLOCKSTATE • CAACOLOCKVAL • CAAELEMNUM • CAAINVLCVI • CAAINVLCVINUM • CAAPARITY For CFLEVEL=4 and higher coupling facilities, the following additional answer area field is returned: • CAADATACACHED For CFLEVEL=5 and higher coupling facilities, the following additional answer area field is returned: • CAAVERSION **Action:** No action is necessary. However, if you expected this structure to have adjunct data, you may want to: • Verify the CONTOKEN is from the correct invocation of IXLCONN. • Check the IXLCONN macro that defined the structure to be sure adjunct data was specified.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the specified storage area for adjunct data (ADJAREA) is not addressable. The adjunct data was not retrieved; however, the entry data was returned in BUFLIST or BUFFER, if requested. The following fields are returned in the answer area: • CAACHANGED • CAACOLOCKSTATE • CAACOLOCKVAL • CAAELEMNUM • CAAINVLCVI • CAAINVLCVINUM • CAAPARITY For CFLEVEL=4 and higher coupling facilities, the following additional answer area field is returned: • CAADATACACHED For CFLEVEL=5 and higher coupling facilities, the following additional answer area field is returned: • CAAVERSION **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro. Correct the address specified by ADJAREA and rerun the request asking for adjunct data only.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The VECTORINDEX specified is not valid. Request processing was suppressed. **Action:** Specify a vector index that is within the current range of the number of vector entries for the local vector requested for this structure. The number of vector entries is determined by the VECTORLEN parameter specified on the IXLCONN macro and may be changed by issuing the IXLVECTR macro.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request with ASSIGN=NO specified failed because the entry designated by NAME is not in the cache structure. **Action:** No action is necessary. However, if this return code and reason code are unexpected, try the following: • Verify that NAME is uncorrupted. • Verify that CONTOKEN is for the correct structure. You might want to issue the request with ASSIGN=YES to allocate a directory entry for this NAME.
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified by STGCLASS exceeds the maximum number of storage classes defined for the cache structure. **Action:** The number of storage classes allowed for a structure are defined on the IXLCONN macro by the NUMSTGCLASS parameter. Correct the STGCLASS parameter to specify a valid storage class.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program Error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data being read. The number of elements in the retrieved entry is returned in the answer area in the field CAAELEMNUM. **Action:** If more space is available, specify a larger buffer size and reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. Allocation of a directory entry and/or data entry was necessary, but was unavailable or could not be reclaimed. The answer area field, CAASTGCLFULL, contains the storage class from which the reclaiming operation failed. **Action:** • Determine if any data items may be cast-out to make room for this item. • Check your usage of storage classes to see if some data items can be moved to a different storage class (preferably with a lower priority) so some entries in the structure can be freed. • Determine if a rebuild of the structure is necessary to make room for more data entries/items.

Table 30. Return and Reason Codes for the IXLCACHE REQUEST=READ_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=READ_DIRINFO

Description

A READ_DIRINFO request allows you to retrieve directory information from the structure and store it in the storage areas specified by BUFLIST or BUFFER. You can specify that all directory entries be processed (CRITERIA=ALL) or only those directory entries that contain changed or locked-for-cast-out data (CRITERIA=CHANGED). You can further filter the selection of entries for processing by specifying NAME and optionally NAMEMASK.

You can request that all directory entry information be returned for the specified data items (DIRINFOFMT=DIRENTRYLIST) or only a subset of the information be returned (DIRINFOFMT=NAMELIST).

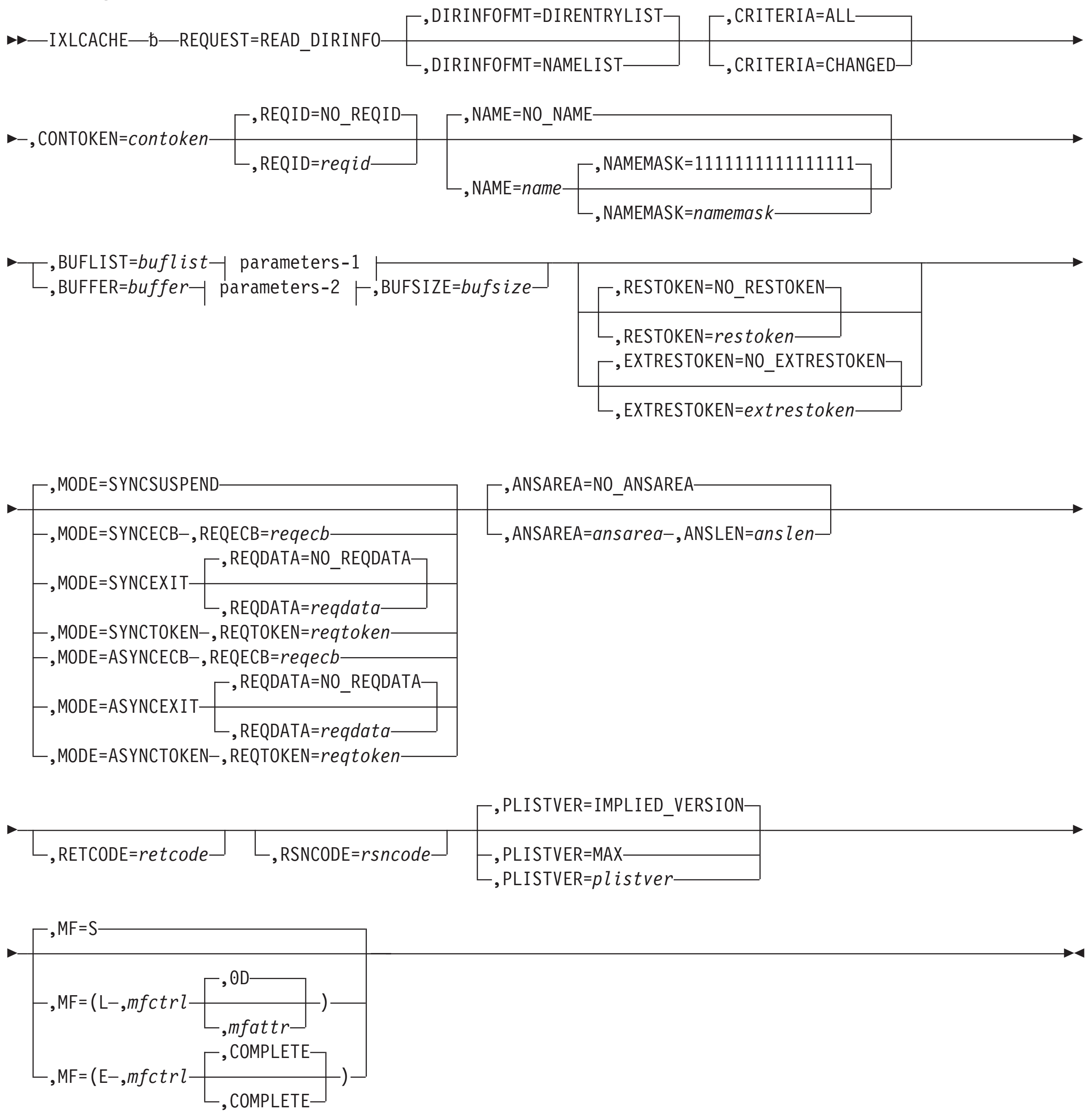
If the request completes prematurely because it exceeds the model-dependent time-out criteria or the specified buffer area (BUFLIST or BUFFER) is full, a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area. The token can be specified on the next READ_DIRINFO request to resume processing with the next directory entry to be processed. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=READ_DIRINFO is as follows:

IXLCACHE Macro

main diagram



parameters-1

```
      ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─────┐
►►────┼────────────────────────────────────────────────────────────────────────────────┼────────►
      │                                           ┌─,BUFALET=NO_BUFALET─┐  ┌─,BUFADDRSIZE=31─┐│
      ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├─────┼─────────────────────┼──┼─────────────────┼┤
      │                                           └─,BUFALET=bufalet────┘  └─,BUFADDRSIZE=64─┘│
      │                          ┌─,BUFADDRSIZE=31─┐                                          │
      └─,BUFADDRTYPE=REAL────────┼─────────────────┼──────────────────────────────────────────┘
                                 └─,BUFADDRSIZE=64─┘

►─,BUFNUM=bufnum──────────────────────────────────────────────────────────────────────────────►◄
```

parameters-2

```
      ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY───────────┐
►►────┼───────────────────────────────────────────────┼───────────────────────────────────────►◄
      │                  ┌─,BUFSTGKEY=CALLERS_KEY─┐   │
      ├─,PAGEABLE=YES────┼────────────────────────┼───┤
      │                  └─,BUFSTGKEY=bufstgkey───┘   │
      └─,PAGEABLE=NO──────────────────────────────────┘
```

Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea*,ANSLEN=*anslen* is required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_DIRINFO are listed in alphabetical order. Default values are underlined:

REQUEST=READ_DIRINFO
Use this input parameter to specify that directory information be retrieved from the directory entries specified by CRITERIA, NAME and NAMEMASK, and placed in the storage areas specified by BUFLIST or BUFFER. The format and content of the directory information returned is specified by the DIRINFOFMT parameter.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The information may include a restart token or extended restart token when the request exceeds the model-dependent time-out criteria or the specified buffer area is filled. The format of the answer area is described by the IXLYCAA mapping macro.

If the request completes successfully, the answer area contains the number of directory entries processed by the request (field CAADIRCOUNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=31
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this output parameter to specify a buffer area to contain the directory information returned from the request.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.
- Does not start below storage address 512.

Use the BUFSIZE description for specifying the size of the buffer.

Note: See the DIRINFOFMT parameter for information about the format of the data returned in the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the data returned from the request.

,BUFLIST=*buflist*
Use this output parameter to specify a list of buffers to hold the directory information returned from the request. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. Cache services treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

Use the BUFNUM parameter to specify the number of buffers in the buffer list.

Note: See the DIRINFOFMT parameter for information about the format of the data returned in the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CRITERIA=ALL
,CRITERIA=CHANGED
Use this input parameter to specify the criteria for selecting which directory entries will be read. (You can further limit the selection by specifying NAME and optionally NAMEMASK.)

ALL
Directory entry information associated with all cached data items will be retrieved.

CHANGED
Directory entry information associated with only those data items that are changed or locked-for-cast-out will be retrieved.

,DIRINFOFMT=DIRENTRYLIST
,DIRINFOFMT=NAMELIST
Use this input parameter to specify what directory information will be returned to the BUFFER area or BUFLIST buffers for each of the specified data items.

DIRENTRYLIST
All the directory entry information is returned. The information returned includes:
- Name
- User data
- Storage class assigned to
- Indication of whether data is currently cached
- Indication of whether data is changed (if cached)
- Castout class assigned to (if changed)
- Parity bits
- Value and stat of the cast-out lock
- Indication of which connected users have registered interest
- Size of the data entry
- For structures allocated in a coupling facility of CFLEVEL=5 or higher, the version number.

See the IXLYDEIB macro for the format of the data returned in the BUFFER area or the BUFLIST buffers.

NAMELIST
A subset of the directory entry information is returned. The information returned includes:
- Name
- User data
- Size of the data entry.

See the IXLYCANB macro for the format of the data returned in the BUFFER area or BUFLIST buffers.

For more information about the format of the returned information, see *z/OS MVS Programming: Sysplex Services Guide* or the mapping macros as described in *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*
Use this input parameter to specify an extended restart token that can be used to resume processing of a READ_DIRINFO request that completed prematurely because it exceeded the model-dependent time-out criteria or because the specified buffer area is full. The extended restart token is returned in the answer area (field CAAEXTRESTOKEN), and should be specified on the next READ_DIRINFO request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria or the buffer does not become full, the extended restart token will not be provided.

Notes:
1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.
3. Specifying an extended restart token requires that the length of the answer area be at least the length of CAALEVEL1LEN.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=NO_NAME
,NAME=*name*
Use this input parameter to filter by name, the data item for which associated directory entry information will be read. You may use this parameter along with the NAMEMASK parameter to select certain data items. See the NAMEMASK parameter for more information on this topic.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NAMEMASK=1111111111111111
,NAMEMASK=*namemask*
Use this input parameter to specify which characters in the name specified by NAME are to be used in selecting data items for processing. This parameter allows you to select multiple data items based on common characters in NAME.

The position of each bit in NAMEMASK corresponds to the same relative character position in NAME. A one indicates that the corresponding letter should be used in selecting entries; a zero indicates that the corresponding letter should not be used.

Specifying a name mask with all zeros causes all names to be selected for processing. Specifying a name mask with all ones causes only the name specified by NAME to be selected.

For more information on how NAMEMASK may be used, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the bit-mask for the name specified on the NAME keyword.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*

Use this input parameter to specify a restart token that can be used to resume processing of a READ_DIRINFO request that completes prematurely because it exceeds the model-dependent time-out criteria or the specified buffer area (BUFLIST or BUFFER) is full. The restart token is returned in the answer area, and should be specified on the next READ_DIRINFO request to resume processing with the next directory entry to be processed.

If the request does not exceed the model-dependent time-out criteria or the buffer does not become full, the returned token will not be provided.

Notes:

1. Specifying a restart token of all zeros causes cache services to treat all of the directory entries as unprocessed.
2. Do not specify a restart token other than the one returned in the answer area or one set to all zeros because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The following information has been returned in the answer area: • The number of elements returned in the BUFFER or BUFLIST area (field CAADIRCOUNT) • A token for restarting the request (field CAARESTOKEN or CAAEXTRESTOKEN). **Action:** Reissue the request using the restart token (RESTOKEN or CAAEXTRESTOKEN). For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** The request completed prematurely because the buffer specified by BUFFER, or the buffers specified by BUFLIST, is full. The following information has been returned in the answer area: • The number of elements returned in the BUFFER or BUFLIST area (field CAADIRCOUNT) • A token for restarting the request (field CAARESTOKEN or CAAEXTRESTOKEN) **Action:** Reissue the request, or increase the size of the buffer(s) and rerun your program. You can reissue the request using the restart token (RESTOKEN or EXTRESTOKEN). **Note:** Process the data that has been returned in the buffers prior to reissuing the request. The data returned from the original request will be overwritten if you specify the same buffer address. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to resume the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request. • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:**Program error. The size of the BUFLIST areas or BUFFER area is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size and reissue the request. Consider using the BUFLIST parameter if BUFFER was specified.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. **Action:**Discard the results of the initial request and reissue the request with an EXTRESTOKEN value of zero. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 31. Return and Reason Codes for the IXLCACHE REQUEST=READ_DIRINFO Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=READ_STGSTATS

Description

A READ_STGSTATS request allows you to retrieve statistical information for a specified storage class (STGCLASS) and store it in the area specified by STGSTATS. The format of the output returned by this request is mapped by IXLYCSCS. For more information on the statistical data returned for storage classes, see *z/OS MVS Programming: Sysplex Services Guide*.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=READ_STGSTATS is as follows:

main diagram

```
                                                                              ┌─,REQID=NO_REQID─┐
►►──IXLCACHE──ƀ──REQUEST=READ_STGSTATS──,STGSTATS=stgstats──,CONTOKEN=contoken─┼─────────────────┼──────►
                                                                              └─,REQID=reqid────┘

                         ┌─,MODE=SYNCSUSPEND──────────────────────────────┐
►─,STGCLASS=stgclass─────┼────────────────────────────────────────────────┼─────────────────────────────►
                         ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤
                         │                ┌─,REQDATA=NO_REQDATA─┐         │
                         ├─,MODE=SYNCEXIT─┼─────────────────────┼─────────┤
                         │                └─,REQDATA=reqdata────┘         │
                         ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
                         ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
                         │                 ┌─,REQDATA=NO_REQDATA─┐        │
                         ├─,MODE=ASYNCEXIT─┼─────────────────────┼────────┤
                         │                 └─,REQDATA=reqdata────┘        │
                         └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┘

   ┌─,ANSAREA=NO_ANSAREA──────────────┐
►──┼──────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬───────────────────►
   └─,ANSAREA=ansarea─,ANSLEN=anslen──┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────┼──────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                ┌─,0D─────┐             │
   └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl─┼─────────┼─)──────────┤
                                  │                └─,mfattr─┘             │
                                  │                ┌─,COMPLETE─┐           │
                                  └─,MF=(E─,mfctrl─┼───────────┼─)─────────┘
                                                   └─,COMPLETE─┘
```

Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* must be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_STGSTATS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_STGSTATS
Use this input parameter to specify that statistical information for the specified storage class (STGCLASS) be returned in the storage area specified by STGSTATS.

,ANSAREA=NO_ANSAREA

,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned by the request will be put.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA

,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STGCLASS=*stgclass*
Use this input parameter to specify the storage class for which statistical information will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage class.

,STGSTATS=*stgstats*

Use this output parameter to specify a storage area to contain the storage class statistics returned from the request. The format of the STGSTATS area is described by mapping macro IXLYCSCS.

For more information about the data returned by this request, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 256-byte area where the storage statistics information will be placed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified by STGCLASS exceeds the maximum number of storage classes defined for the cache structure. **Action:** The number of storage classes allowed for a structure are defined on the IXLCONN macro by the NUMSTGCLASS parameter. Correct the STGCLASS parameter to specify a valid storage class.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0869	**Equate Symbol:** IXLRSNCODEBADSTGSTATS **Meaning:** Program error. The storage area specified by STGSTATS is not addressable. **Action:** Ensure that: • The address passed in STGSTATS is valid. • The STGSTATS area was not previously freed. • If the caller is running in AR-mode and STGSTATS was specified using explicit register notation, ensure that the corresponding access register was updated appropriately. • If the caller is disabled, then STGSTATS must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXLCACHE macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.

Table 32. Return and Reason Codes for the IXLCACHE REQUEST=READ_STGSTATS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=REG_NAMELIST

Description

A REG_NAMELIST request allows you to specify a list of data items for which you want to register interest. Each data item is represented by a registration block, mapped by the IXLYCRRB macro. The registration block contains information identifying the data item, its storage class, its vector index, whether to assign a directory entry, and whether the vector index is being reassigned to the current entry from some other entry in which interest will be deregistered. The registration blocks are contained in the storage area specified by BUFFER.

The system references the registration blocks by an index into the buffer and processes them sequentially beginning with the registration block identified by the first index (STARTINDEX) and ending with the data item identified by the last index (ENDINDEX).

The system returns state information for each processed data item represented by a registration block. The information is returned in the area specified by NSBAREA, and is mapped by the IXLYNSB macro.

A REG_NAMELIST request can complete prematurely because of a failure processing one of the registration blocks in the list or because the request exceeds the time-out criteria for the coupling facility. (Time-out criteria is model-dependent.) When a request completes prematurely, the system returns an index value (CAARNLINDEX) in the answer area which you can use to restart the REG_NAMELIST request.

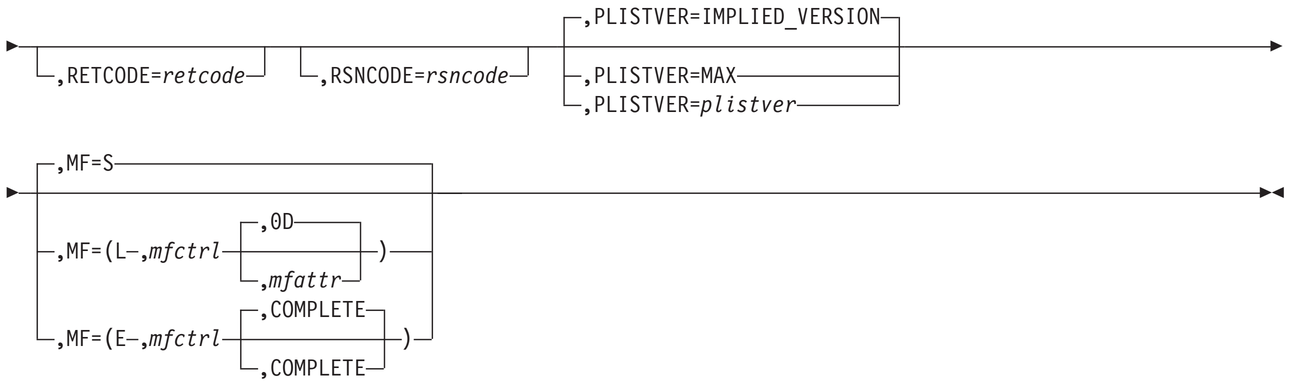
A REG_NAMELIST request can be issued only for cache structures allocated in a coupling facility of CFLEVEL=2 or higher. REG_NAMELIST requests issued for a cache structure allocated in a coupling facility of CFLEVEL=0 or 1 will fail.

Syntax Diagram

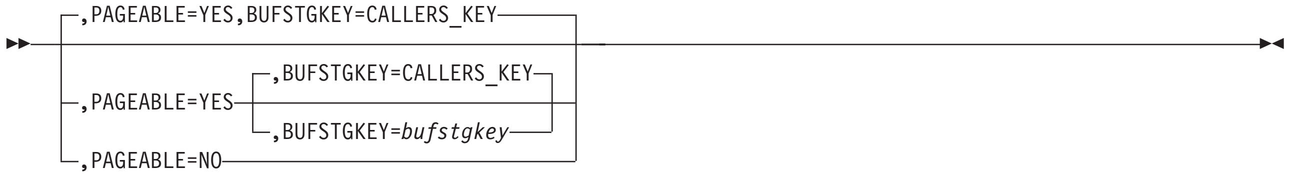
The syntax for the IXLCACHE REQUEST=REG_NAMELIST is as follows:

main diagram

```
►►──IXLCACHE──␢──REQUEST=REG_NAMELIST──,STARTINDEX=startindex──,ENDINDEX=endindex──,NSBAREA=nsbarea──────►


                     ┌─,REQID=NO_REQID─┐
►─,CONTOKEN=contoken─┼─────────────────┼─,BUFFER=buffer─┤ parameters-1 ├─,BUFSIZE=bufsize──────────────►
                     └─,REQID=reqid────┘

  ┌─,MODE=SYNCSUSPEND─────────────────────────┐  ┌─,ANSAREA=NO_ANSAREA──────────────┐
►─┼───────────────────────────────────────────┼──┼───────────────────────────────────┼──────────────────►
  ├─,MODE=SYNCECB─,REQECB=reqecb──────────────┤  └─,ANSAREA=ansarea─,ANSLEN=anslen─┘
  │                ┌─,REQDATA=NO_REQDATA─┐    │
  ├─,MODE=SYNCEXIT─┼─────────────────────┼────┤
  │                └─,REQDATA=reqdata────┘    │
  ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken────────┤
  ├─,MODE=ASYNCECB─,REQECB=reqecb─────────────┤
  │                 ┌─,REQDATA=NO_REQDATA─┐   │
  ├─,MODE=ASYNCEXIT─┼─────────────────────┼───┤
  │                 └─,REQDATA=reqdata────┘   │
  └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken───────┘
```



parameters-1



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea* and ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=REG_NAMELIST are listed in alphabetical order. Default values are underlined:

REQUEST=REG_NAMELIST
Use this input parameter to specify that you want to register interest in a list of data items. Each data item is represented by a registration block, mapped by the IXLYCRRB macro, stored in the area specified by BUFFER.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the answer area information returned from the request will be stored.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFFER=*buffer*
Use this input parameter to specify a buffer area to contain an array of from 1 to 32 registration blocks that define the data items that are to be registered. The BUFFER area must be addressable from your primary address space or from your PASN access list.

The format of the registration block is described by mapping macro IXLYCRRB.

You can define the buffer size to be a total of up to 65536 bytes, but it should not be larger than what you actually require to hold the maximum number of registration blocks, subject to the other buffer length requirements stated below.

Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer. See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the registration blocks.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer are in the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage key in the format kkkkxxxx, where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,ENDINDEX=*endindex*
Use this input parameter to specify the ending index for registration block processing. The index value must be greater than or equal to the value specified for STARTINDEX. Valid values are 1 - 32.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field containing the ending index value.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**

,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NSBAREA=*nsbarea*
Use this output parameter to specify a storage area to contain name state information after the system has processed the registration blocks specified in the BUFFER area. The NSBAREA area must be addressable from your primary address space or from your PASN access list.

The format of the NSBAREA area is described by mapping macro IXLYNSB.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 256-byte area to contain the name state information.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage area specified by BUFFER is in pageable or potentially pageable storage, and, if the area could become pageable, designate responsibility for ensuring that the area remains page-fixed for the duration of the request.

PAGEABLE=YES
Specify this option when you are using any one of the following types of storage and you want the system to ensure that the storage becomes or remains page-fixed for the duration of the request:
- Pageable storage
- Disabled reference (DREF) storage
- Page-fixed storage that could become pageable before the request completes processing.

PAGEABLE=NO
Specify this option when using any one of the following types of storage:
- Fixed storage
- Page-fixed storage that you will ensure remains page-fixed for the duration of the request

If you specify PAGEABLE=NO and the request processes asynchronously, you must keep the storage fixed until you are sure the request has completed. (The MODE value specified determines how you are notified of request completion.) See *z/OS MVS Programming: Sysplex Services Guide* for these guidelines.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*

Use this input parameter to specify the starting index for registration block processing. Valid STARTINDEX values are from 1 to the value of ENDINDEX. The first registration block has index number 1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field containing the starting index value.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the next registration block to be processed has been returned in the answer area (field CAARNLINDEX). **Action:** Process the entries that have completed on the current request and then reissue the request. The entries that were processed are indexed from STARTINDEX to CAARNLINDEX-1. To restart the request, update STARTINDEX with the value of CAARNLINDEX, the index of the next name element to be processed. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The vector index in the registration block indexed by CAARNLINDEX is not valid. None of the registration blocks have been processed. All vector indexes for all registration blocks in the buffer have been set to indicate that the local buffer is not valid. **Action:** Correct the vector index value and reissue the REG_NAMELIST request with the same STARTINDEX and ENDINDEX values.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. A REG_NAMELIST registration block specified a storage class value that exceeded the maximum defined storage class for the structure. CAARNLINDEX contains the index of the registration block that contained the storage class value in error. All registration blocks preceding this block were processed. **Action:** Correct the storage class value in the registration block indexed by CAARNLINDEX and reissue the REG_NAMELIST request starting with that registration block.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If the caller is running in AR-mode and the BUFFER parameter was specified using explicit register notation, the corresponding access register was updated appropriately.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER is not addressable. All bits in the local cache vector may be reset to overindicate that the data items are not valid. **Action:** Ensure that: • The correct buffer address was used for BUFFER. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • The buffer area is addressable in the caller's primary address space or from the caller's PASN access list. • If the caller is running in AR-mode and the BUFFER parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • Requirements for BUFFER and BUFSIZE. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0874	**Equate Symbol:** IXLRSNCODEBADRNLINDEX **Meaning:** Program error. Either the value specified for either STARTINDEX or ENDINDEX is not valid or the size of the buffer specified by BUFFER is smaller than required based on the value of ENDINDEX. **Action:** Ensure that: • The values specified by STARTINDEX and ENDINDEX are in the range of 1 to 32. • The value specified by ENDINDEX is greater than or equal to the value of STARTINDEX. • The value specified by ENDINDEX does not imply a larger BUFFER size than was actually specified on the REG_NAMELIST request.
8	xxxx0875	**Equate Symbol:** IXLRSNCODEBADNSBAREA **Meaning:** Program error. The storage area specified by NSBAREA is not addressable. **Action:** Ensure that: • The address passed in NSBAREA is valid. • The NSBAREA area was not previously freed. • The NSBAREA area is addressable from the caller's primary address space or from the caller's PASN access list. • If the caller is running in AR-mode and NSBAREA was specified using explicit register notation, ensure that the corresponding access register was updated appropriately. • If the caller is disabled, then NSBAREA must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXLCACHE macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. Allocation of a directory entry was necessary, but was unavailable or could not be reclaimed. In the answer area, CAASTGCLFULL contains the storage class from which the reclaiming operation failed. CAARNLINDEX contains the index of the registration block that was being processed when the error occurred. All prior registration blocks were processed. **Action:** • Determine if any data items may be cast-out to make room for this item. • Check your usage of storage classes to see if some data items can be moved to a different storage class (preferably with a lower priority) so some entries in the structure can be freed. • Determine if a rebuild or an alter of the structure is necessary to make room for more data entries/items. After correcting the error, restart the REG_NAMELIST request starting with the registration block indexed by CAARNLINDEX.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.

Table 33. Return and Reason Codes for the IXLCACHE REQUEST=REG_NAMELIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either continue processing using single READ_DATA requests to perform the function of the REG_NAMELIST request or disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=2 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=RESET_REFBIT

Description

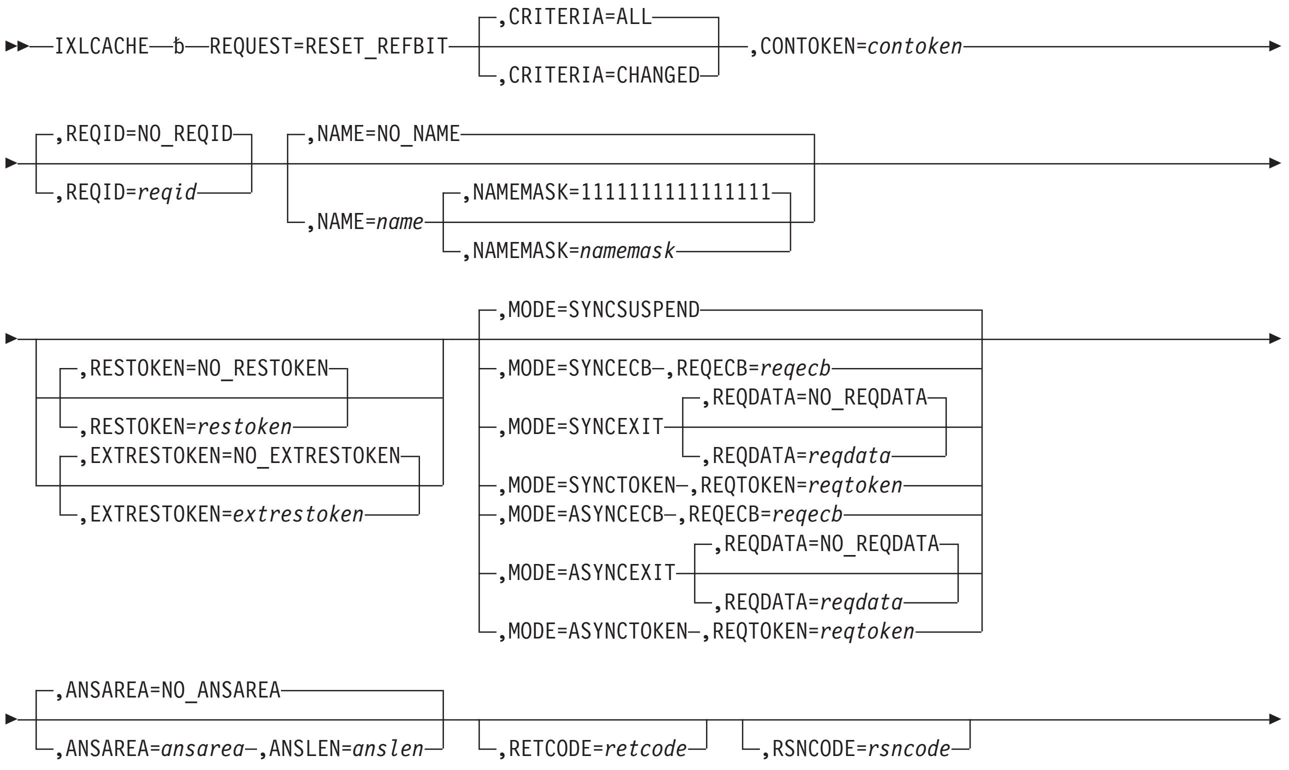
A RESET_REFBIT request allows you to reset the reference bit in the directory entries for the specified cached data items to indicate that the data items were not recently referenced. You can specify that all directory entries in the structure are processed (CRITERIA=ALL) or only those that contain changed or locked-for-cast-out data (CRITERIA=CHANGED). You can further filter the selection of entries to be processed by specifying NAME and NAMEMASK. The total number of processed directory entries and the number of these directory entries for which the reference bit was actually reset are both returned in the answer area (ANSAREA).

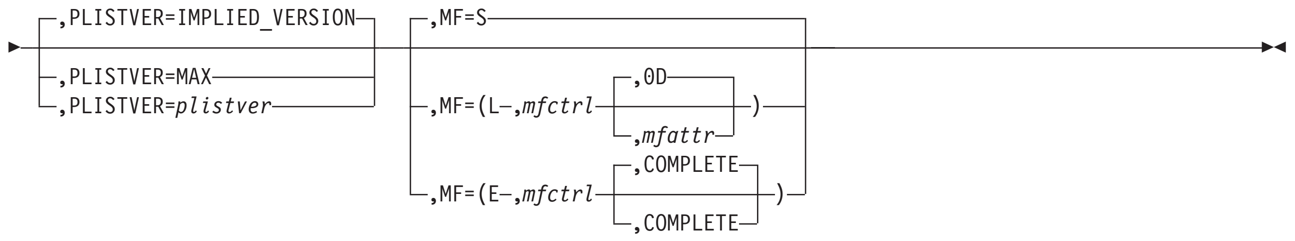
If the request exceeds the model-dependent time-out criteria before processing completes, either a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area. The token can be specified on the next RESET_REFBIT request to resume processing with the next data item to be processed. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=RESET_REFBIT is as follows:

main diagram





Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea*,ANSLEN=*anslen* is required.

Parameter Descriptions

The parameter descriptions for REQUEST=RESET_REFBIT are listed in alphabetical order. Default values are underlined:

REQUEST=RESET_REFBIT
Use this input parameter to specify that the directory entry reference bit for the entries specified by CRITERIA, NAME, and NAMEMASK be reset to indicate that entries are unreferenced. The total number of directory entries processed and the number of these directory entries that had their reference bit reset are returned in the answer area.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro. The following information is returned in the answer area:

- The number of directory entries processed by the request (field CAADIRCOUNT).
- The number of processed directory entries for which the reference bit was reset (field CAAREFCOUNT).
- See the RESTOKEN and EXTRESTOKEN parameters for a description of the restart token.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request is to be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CRITERIA=ALL

,CRITERIA=CHANGED

Use this input parameter to specify the criteria for selecting which directory entries will be processed. (You can further limit the selection by specifying NAME with or without NAMEMASK.)

ALL

The reference bit will be reset in all allocated directory entries.

CHANGED

The reference bit will be reset in only those directory entries that contain changed or locked-for-cast-out data.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*

Use this input parameter to specify an extended restart token that can be used to resume processing of a RESET_REFBIT request that exceeded the model-dependent time-out criteria. The extended restart token is returned in the answer area (field CAAEXTRESTOKEN), and should be specified on the next RESET_REFBIT request to resume processing with the next data item to be processed.

If the request does not exceed the model-dependent time-out criteria, the extended restart token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes cache services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.
3. Specifying an extended restart token requires that the length of the answer area be at least the length of CAALEVEL1LEN.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=NO_NAME
,NAME=*name*
Use this input parameter to filter by name the data items for which the associated directory entry reference bit will be reset. You may use this parameter along with the NAMEMASK parameter to selectively filter out data items. See the NAMEMASK parameter for more information on this option.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NAMEMASK=1111111111111111
,NAMEMASK=*namemask*
Use this input parameter to specify which characters in the name specified by NAME are to be used in selecting data items for processing. This parameter allows you to select multiple data items based on common characters in NAME.

The position of each bit in NAMEMASK corresponds to the same relative character position in NAME. A one indicates that the corresponding letter should be used in selecting entries; a zero indicates that the corresponding letter should not be used.

Specifying a name mask with all zeros causes all names to be selected for processing. Specifying a name mask with all ones causes only the name specified by NAME to be selected.

For more information on how NAMEMASK may be used, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the bit-mask for the name specified on the NAME keyword.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID

,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*

Use this input parameter to specify a restart token that can be used to resume processing of a RESET_REFBIT request that exceeded the model-dependent time-out criteria. The restart token is returned in the answer area and should be specified on the next RESET_REFBIT request to resume processing with the next directory entry to be processed.

If the request does not exceed the model-dependent time-out criteria, the returned token will not be provided.

Notes:

1. Specifying a restart token of all zeros causes cache services to treat all of the directory entries as unprocessed.
2. Do not specify a restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. A token for restarting the request has been returned in the answer area (field CAARESTOKEN or CAAEXTRESTOKEN). The number of processed directory entries that initially had the reference bit set (field CAAREFCOUNT), and the number of directory entries processed (field CAADIRCOUNT) are returned in the answer area. **Action:** Reissue the request using the appropriate restart token. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to resume the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request. • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. **Action:**Discard the results of the initial request and reissue the request with an EXTRESTOKEN value of zero. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 34. Return and Reason Codes for the IXLCACHE REQUEST=RESET_REFBIT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=SET_RECLVCTR

Description

A SET_RECLVCTR request allows you to define, activate, or deactivate a reclaiming vector (RECLVCTR) for a specified storage class (STGCLASS). Once defined, the reclaiming vector specifies how many reclaims are allowed against each storage class defined in the structure for requests targeted to the storage class to which the reclaim vector pertains. The reclaiming vector is associated with the specified storage class (STGCLASS) and will be used to satisfy subsequent IXLCACHE requests that require resources in that storage class.

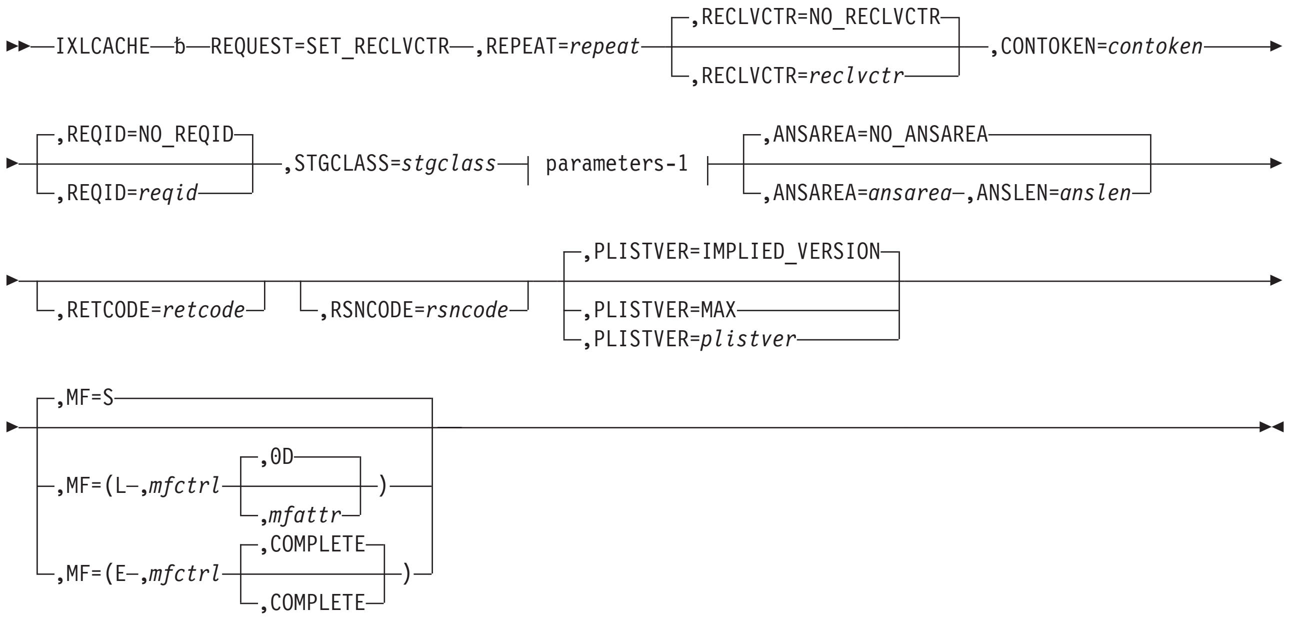
You can also specify the number of passes (REPEAT) through the reclaim vector before reclaim processing reverts to the default reclaim algorithm. (By default, the system attempts to reclaim the least recently used resources belonging to data items in the target storage class.) The reclaiming vector you specify remains active for the specified number of passes, or until it is deactivated. Specifying a zero on the REPEAT parameter deactivates the reclaiming vector.

When the system processes an IXLALTER request for a cache structure, all active reclaim vectors associated with all storage classes for the structure are deactivated. While the alter process continues, the system rejects any attempt to activate a reclaim vector. When the alter process completes, the system does not automatically reactivate any reclaim vectors that were deactivated when the structure alter was initiated. It is the responsibility of the user to activate any new or still-needed reclaim vectors.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=SET_RECLVCTR is as follows:

main diagram



parameters-1

```
     ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►►───┼────────────────────────────────────────────┼─────────────────────────────────────────────►◄
     ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
     │                 ┌─,REQDATA=NO_REQDATA─┐    │
     ├─,MODE=SYNCEXIT──┼─────────────────────┼────┤
     │                 └─,REQDATA=reqdata────┘    │
     ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
     ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
     │                  ┌─,REQDATA=NO_REQDATA─┐   │
     ├─,MODE=ASYNCEXIT──┼─────────────────────┼───┤
     │                  └─,REQDATA=reqdata────┘   │
     ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┤
     └─,MODE=ASYNCNORESPONSE──────────────────────┘
```

Note: When MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=SET_RECLVCTR are listed in alphabetical order. Default values are underlined:

REQUEST=SET_RECLVCTR
Use this input parameter to specify that the reclaiming vector (RECLVCTR) for the specified storage class (STGCLASS) be activated or deactivated.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned by the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=<u>S</u>
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,<u>0D</u>)**
,MF=(E,*mfctrl***)**

,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

MODE=SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,RECLVCTR=NO_RECLVCTR
,RECLVCTR=*reclvctr*
Use this input parameter to define and activate a reclaiming vector. The RECLVCTR area must be arranged as follows:

- The area must contain, starting at offset zero, a fully initialized array of halfwords, each corresponding to a storage class. (There must be one halfword for each storage class defined for the structure.) The relative position of a halfword in the array determines which storage class that halfword is associated with. For instance, the first halfword in the array is associated with storage class one, the second halfword is associated with storage class two, and so forth.
- Each halfword in the array must contain the reclaiming count for its associated storage class. You can avoid reclaims from a storage class by placing zero in the appropriate halfword.

If the specified reclaiming vector is not already active and a REPEAT factor of zero is specified, the REPEAT factor is ignored. If the reclaiming vector is already active, a zero REPEAT factor deactivates the reclaiming vector.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a storage area (with a length of double the number of storage classes defined for the structure on the IXLCONN macro, keyword NUMSTGCLASS) that contains the reclaiming vectors.

,REPEAT=*repeat*
Use this input parameter to specify the repeat factor for the reclaiming vector being activated (or deactivated). The repeat factor defines the number of passes through the reclaiming vector before the

vector deactivates. The specified value must fall within the range 0 to 65535, inclusive. Specifying a repeat factor of zero causes the active reclaiming vector for the specified storage class to be deactivated.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the repeat factor.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STGCLASS=*stgclass*
Use this input parameter to specify the storage class for which a reclaim vector is to be activated or deactivated.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage class.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed. • If you specified MODE=ASYNCNORESPONSE, no action is required. You will not be notified when the request completes.

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0414	**Equate Symbol:** IXLRSNCODERCLVCTRNOTSET **Meaning:** The reclaim vector was not set because either the structure size or the entry-to-element ratio is being changed through IXLALTER. **Action:** Set the reclaim vector when notified that the alter processing for the structure is complete.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified by STGCLASS exceeds the maximum number of storage classes defined for the cache structure. **Action:** The number of storage classes allowed for a structure are defined on the IXLCONN macro by the NUMSTGCLASS parameter. Correct the STGCLASS parameter to specify a valid storage class.
8	xxxx0832	**Equate Symbol:** IXLRSNCODENORCLVCTR **Meaning:** Program error. The REPEAT value specified is greater than zero, but the reclaim vector storage area (RECLVCTR) was not specified. **Action:** If you intend to deactivate an active reclaim vector, specify zero for REPEAT. If you intend to specify a repeat factor for a reclaim vector you are activating, specify a valid reclaim vector as described under the RECLVCTR parameter.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0868	**Equate Symbol:** IXLRSNCODEBADRECLVCTR **Meaning:** Program error. The storage area specified by RECLVCTR is not addressable. **Action:** Ensure that: • The address specified by RECLVCTR is valid. • The RECLVCTR area was not previously freed. • If the caller is running in AR-mode and the RECLVCTR parameter was specified using explicit register notation, ensure that the corresponding access register was updated appropriately. • If the caller is disabled, then RECLVCTR must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, you specified SYSSTATE ASCENV=AR before issuing the IXLCACHE macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 35. Return and Reason Codes for the IXLCACHE REQUEST=SET_RECLVCTR Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=UNLOCK_CASTOUT

Description

An UNLOCK_CASTOUT request allows you to release the cast-out locks for multiple data items that are named in the storage areas specified by either BUFFER or BUFLIST. The data items listed by name in the buffer(s) are referenced by an index into the buffer(s), and are processed sequentially beginning with the data item identified by the first index (FIRSTNAME) and ending with the data item identified by the last index (LASTNAME). Besides releasing cast-out locks, the request updates the parity and user data in the directory entry for each data item named in the list. The parity and user data for each data item should also be supplied in the buffer(s). (See mapping macro IXLYCUNB for how the buffer(s) should be arranged.)

Once the cast-out lock is released for a data item, the data item might or might not remain associated with its cast-out class:

- If the data entry contains unchanged data, the data item is disassociated from any previously specified cast-out class.
- If the data entry contains changed data (the entry could have had changed data written to it while the cast-out lock was held), the data item is left associated with the last specified cast-out class.

You can also specify that a currently unchanged data item remain associated with its cast-out class by overriding its changed indicator in the directory entry for the data item to indicate the data item is changed. This information should also be supplied in the buffer(s).

An UNLOCK_CASTOUT request can either fail or complete prematurely for the following reasons:

- A data item named in the buffer(s) does not exist in the structure. The index of the failing name is returned in the answer area (ANSAREA). All names preceding the failing name are processed.
- You do not hold the cast-out lock for a data item named in the buffer(s). The index of the failing name and the value of the cast-out lock are returned in the answer area because of either a user or PROCESSID mismatch. All names preceding the failing name are processed.
- The request has exceeded the model-dependent time-out criteria. The index of the next name to be processed is returned in the answer area. All names preceding this name are processed.

The index value of the data item name that caused the problem is returned in CAAULINDEX. You may update FIRSTNAME to point after CAAULINDEX, and reissue the request.

Syntax Diagram

The syntax diagram for the IXLCACHE REQUEST=UNLOCK_CASTOUT is as follows:

IXLCACHE Macro

main diagram

```
►►──IXLCACHE──b──REQUEST=UNLOCK_CASTOUT──,FIRSTNAME=firstname──,LASTNAME=lastname──────────────►

   ┌─,PROCESSID=NO_PROCESSID─┐                  ┌─,REQID=NO_REQID─┐
►──┼─────────────────────────┼─,CONTOKEN=contoken─┼─────────────────┼──────────────────────────────►
   └─,PROCESSID=processid────┘                  └─,REQID=reqid────┘

                                                         ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►──┬─,BUFLIST=buflist─┤ parameters-1 ├──────────────────┬─┼────────────────────────────────────────────┼──►
   └─,BUFFER=buffer─┤ parameters-2 ├─,BUFSIZE=bufsize───┘ ├─,MODE=SYNCECB──,REQECB=reqecb──────────────┤
                                                         │                ┌─,REQDATA=NO_REQDATA─┐      │
                                                         ├─,MODE=SYNCEXIT─┼─────────────────────┼──────┤
                                                         │                └─,REQDATA=reqdata────┘      │
                                                         ├─,MODE=SYNCTOKEN──,REQTOKEN=reqtoken─────────┤
                                                         ├─,MODE=ASYNCECB──,REQECB=reqecb──────────────┤
                                                         │                 ┌─,REQDATA=NO_REQDATA─┐     │
                                                         ├─,MODE=ASYNCEXIT─┼─────────────────────┼─────┤
                                                         │                 └─,REQDATA=reqdata────┘     │
                                                         └─,MODE=ASYNCTOKEN──,REQTOKEN=reqtoken────────┘

   ┌─,ANSAREA=NO_ANSAREA──────────────────┐
►──┼──────────────────────────────────────┼──┬────────────────────┬──┬────────────────────┬──────────►
   └─,ANSAREA=ansarea──,ANSLEN=anslen─────┘  └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────┼──────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                     ┌─,0D──────┐   │
   └─,PLISTVER=plistver────────┘  ├─,MF=(L──,mfctrl─────┼──────────┼─)─┤
                                  │                     └─,mfattr──┘   │
                                  │                     ┌─,COMPLETE─┐  │
                                  └─,MF=(E──,mfctrl─────┼───────────┼─)┘
                                                        └─,COMPLETE─┘
```

parameters-1

```
     ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─────────┐
►►───┼─────────────────────────────────────────────────────────────────────────────────┼──────►
     │                                             ┌─,BUFALET=NO_BUFALET─┐ ┌─,BUFADDRSIZE=31─┐ │
     ├─,BUFADDRTYPE=VIRTUAL──┤ parameters-2 ├──────┼─────────────────────┼─┼─────────────────┼─┤
     │                                             └─,BUFALET=bufalet────┘ └─,BUFADDRSIZE=64─┘ │
     │                     ┌─,BUFADDRSIZE=31─┐                                                 │
     └─,BUFADDRTYPE=REAL───┼─────────────────┼─────────────────────────────────────────────────┘
                           └─,BUFADDRSIZE=64─┘

►──,BUFNUM=bufnum────────────────────────────────────────────────────────────────────────────►◄
```

parameters-2

```
     ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY─────────┐
►►───┼─────────────────────────────────────────────┼──────────────────────────────────────────►◄
     │                  ┌─,BUFSTGKEY=CALLERS_KEY─┐ │
     ├─,PAGEABLE=YES────┼────────────────────────┼─┤
     │                  └─,BUFSTGKEY=bufstgkey───┘ │
     └─,PAGEABLE=NO────────────────────────────────┘
```

Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=UNLOCK_CASTOUT are listed in alphabetical order. Default values are underlined:

REQUEST=UNLOCK_CASTOUT
Use this input parameter to specify that the cast-out locks held by your connection for the data items named in the storage areas specified by BUFFER or BUFLIST be released. Additionally, the user data and parity supplied in the buffer(s) updates the existing directory entry information for each data item named in the buffer(s).

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request if the request does not complete successfully. See the return and reason code descriptions for this request to determine what fields in the answer area are valid for non-zero return codes. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=<u>31</u>
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=<u>VIRTUAL</u>
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=<u>NO_BUFALET</u>

,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to contain a list of entries for which cast-out locks should be released.

You must ensure that the storage area specified by BUFFER:

- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.
- Does not start below storage address 512.
- Consists of 32-byte elements, starting at offset zero. See the IXLYCUNB mapping macro for a mapping of the 32-byte element for an UNLOCK_CASTOUT request.

  See the BUFSIZE description for specifying the size of the buffer.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry names to be processed.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold a list of entries for which cast-out locks should be released. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.
- Not start below storage address 512.
- Consist of 32-byte elements, starting at offset zero. See the IXLYCUNB mapping macro for a mapping of the 32-byte element for an UNLOCK_CASTOUT request.

Note: The buffers do not have to be contiguous in storage. Cache services treat BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM parameter to specify the number of buffers in the buffer list.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte storage area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no cast-out locks will be released.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0-16) in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,FIRSTNAME=*firstname*
Use this input parameter to specify an index into the list of names stored in the BUFFER area or BUFLIST buffers. The index you select identifies the first data item to be processed.

FIRSTNAME can be used on a subsequent UNLOCK_CASTOUT request to identify the data item with which processing should resume should the previous request complete prematurely.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the index.

,LASTNAME=*lastname*
Use this input parameter to specify an index into the list of names stored in the BUFFER area or BUFLIST buffers. The index you select identifies the last data item to be processed, and must be greater than, or equal to, the index specified for FIRSTNAME.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the index.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an

asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,PROCESSID=NO_PROCESSID
,PROCESSID=*processid*
Use this input parameter to specify a user-defined process identifier to be compared with the cast-out lock along with the connection identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the user-defined process identifier.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the next name element to be processed has been returned in the answer area (field CAAULINDEX). **Action:** Reissue the request. Update FIRSTNAME (from CAAULINDEX) with the index of the next name element to be processed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx081E	**Equate Symbol:** IXLRSNCODEBADCOLOCKSTATE **Meaning:** You requested that an entry in BUFFER or BUFLIST have its change bit overridden to indicate that the entry contains changed data, but this option is incompatible with the current cast-out lock state value, ″write with castout.″ The ″write with castout″ state is entered when the cast-out lock is obtained by a WRITE_DATA request specifying GETCOLOCK=YES. The following fields are returned in the answer area: • CAACOLOCKSTATE • CAAULINDEX • CAACOLOCKVAL For a description of cast-out lock state values, see mapping macro IXLYCAA. **Action:** The lock state and value are returned in the answer area (ANSAREA). • Determine who holds the lock for this data item. • Verify the names in the buffer you have passed. • Verify the FIRSTNAME and LASTNAME indexes. • Determine why this entry was in your list. You can bypass the current entry by updating FIRSTNAME with the index after the value in CAAULINDEX, and resubmitting your request.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because a name element specified an entry name that is not in the cache structure. All name elements before the offending name were processed; no name elements beyond the offending name were processed. The index of the offending name is returned in the answer area (field CAAULINDEX). **Action:** Reissue the request specifying for FIRSTNAME the next valid name element to be processed.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0829	**Equate Symbol:** IXLRSNCODEBADUNLOCKVAL **Meaning:** Program error. The UNLOCK_CASTOUT request encountered a name element for which the cast-out lock was not held for the connection specified by CONTOKEN and/or the PROCESSID specified. Processing was halted with this name element. The lock state (field CAACOLOCKSTATE) and lock value (field CAACOLOCKVAL) for the element as well as the index value (field CAAULINDEX) for the name element that caused the failure are returned in the answer area (ANSAREA). **Action:** Specify for FIRSTNAME the index of the next name element (CAAULINDEX + 1) for which the cast-out lock is held, and reissue the request.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The name index specified by either FIRSTNAME or LASTNAME is not valid. Some elements may have been processed. CAAULINDEX contains the index of the name that caused the failure. **Action:** Ensure that FIRSTNAME and LASTNAME specify valid indexes into the elements stored in BUFLIST or BUFFER.
8	xxxx082F	**Equate Symbol:** IXLRSNCODEBADPARITY **Meaning:** Program error. The parity bits in a name element are not valid. **Action:** The buffers specified in the request contained parity bits that were not valid. The answer area field CAAULINDEX, contains the index of the name element that caused the failure. The value for the parity bits may be 00, 01, or 11 in bits 2 and 3 of the specification (xxpp xxxx where pp is the parity bits). See mapping macro IXLYCUNB to find the parity bits in the buffer.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the caller's PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information).
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 36. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CASTOUT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=UNLOCK_CO_NAME

Description

An UNLOCK_CO_NAME request allows you to release a single cast-out lock for a data item named in the storage area specified by CUNBAREA. You can also use the request to update the user data in the directory entry for the data item named. The user data, as well as a changed indicator and a parity value, is supplied in the CUNBAREA storage area. (CUNBAREA is mapped by the macro IXLYCUNB.) REQUEST=UNLOCK_CO_NAME is valid only for cache structures allocated in a coupling facility with CFLEVEL=4 or higher.

Once the cast-out lock is released for the data item, the data item might or might not remain associated with its cast-out class:

- If the data entry contains unchanged data, the data item is disassociated from any previously specified cast-out class.
- If the data entry contains changed data (the entry could have had changed data written to it while the cast-out lock was held), the data item is left associated with the last specified cast-out class.

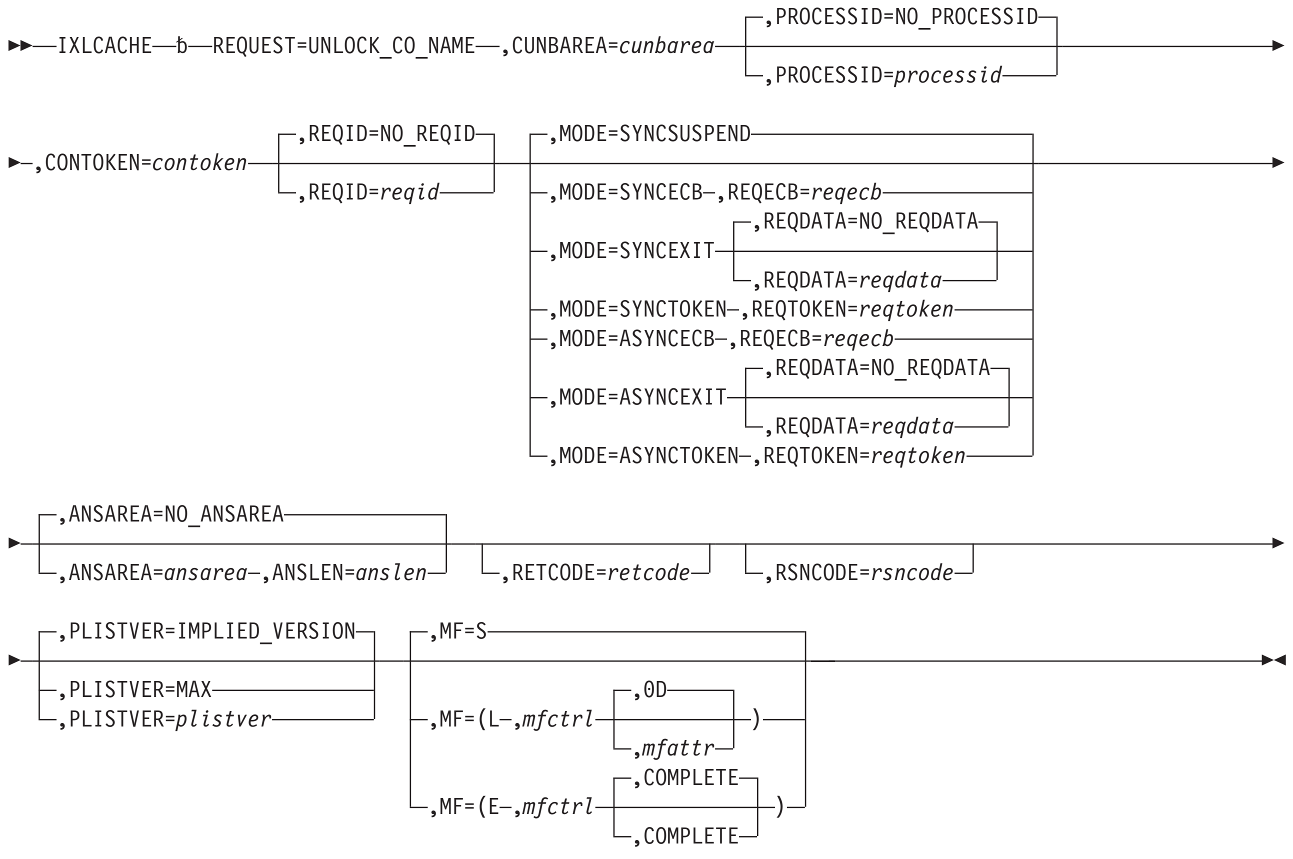
You can specify that a currently unchanged data item remain associated with its cast-out class by overriding its changed indicator in the directory entry for the data item.

Use the UNLOCK_CO_NAME request type to reset a single cast-out lock; use the UNLOCK_CASTOUT request type to reset multiple cast-out locks with a single request.

Syntax Diagram

The syntax diagram for the IXLCACHE REQUEST=UNLOCK_CO_NAME is as follows:

main diagram



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=UNLOCK_CO_NAME are listed in alphabetical order. Default values are underlined:

REQUEST=UNLOCK_CO_NAME
Use this input parameter to specify that the cast-out lock held by your connection for the data item named in the storage area specified by CUNBAREA be released. Additionally, the user data supplied in the CUNBAREA updates the existing directory entry information for the data item.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request when it completes. See the return and reason code descriptions for this request to determine what fields in the answer area are valid for non-zero return codes. The format of the answer area is described by the IXLYCAA mapping macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CUNBAREA=*cunbarea*

Use this input parameter to specify the 32-byte field containing the structure data name followed by request control information for the cast-out lock that is to be released. The format of the 32-byte area is described by the IXLYCUNB macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte area that contains the structure entry name and control information for the UNLOCK_CO_NAME request.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXLCACHE Version Support” on page 248 for a description of the options available with PLISTVER.

,PROCESSID=NO_PROCESSID
,PROCESSID=*processid*

Use this input parameter to specify a user-defined process identifier to be compared with the cast-out lock along with the connection identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the user-defined process identifier.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS Programming: Sysplex Services Guide* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro

Hexadecimal Return Code	Hexadecimal Reason Cod	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection’s complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection’s completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller’s primary address space or from the caller’s PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx081E	**Equate Symbol:** IXLRSNCODEBADCOLOCKSTATE **Meaning:** You requested that an entry named in CUNBAREA have its change bit overridden to indicate that the entry contains changed data, but this option is incompatible with the current cast-out lock state value, ″write with castout.″ The ″write with castout″ state is entered when the cast-out lock is obtained by a WRITE_DATA request specifying GETCOLOCK=YES. The following fields are returned in the answer area: • CAACOLOCKSTATE • CAACOLOCKVAL For a description of cast-out lock state values, see mapping macro IXLYCAA. **Action:** The lock state and value are returned in the answer area (ANSAREA). • Determine who holds the lock for this data item. • Verify the name in the CUNBAREA you have passed.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because the entry name identified in CUNBAREA specified an entry name that is not in the cache structure. **Action:** Reissue the request specifying in CUNBAREA the valid entry name to be processed.
8	xxxx0829	**Equate Symbol:** IXLRSNCODEBADUNLOCKVAL **Meaning:** Program error. The entry name identified in CUNBAREA did not have the castout lock held for the connection specified by CONTOKEN and/or PROCESSID. The lock state (field CAACOLOCKSTATE) and lock value (field CAACOLOCKVAL) for the element are returned in the answer area (ANSAREA). **Action:** Verify that the CONTOKEN and PROCESSID specifications are valid.
8	xxxx082F	**Equate Symbol:** IXLRSNCODEBADPARITY **Meaning:** Program error. The parity value specified in the write-operation-block was not valid. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:** The value specified in the request contained parity bits that were not valid. The value for the parity bits may be 00, 01, or 11 in bits 2 and 3 of the PARITY specification (xxpp xxxx where pp is the parity bits).
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Cod	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either continue processing using an UNLOCK_CASTOUT request to perform the function of the UNLOCK_CO_NAME request or disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=4 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.

Table 37. Return and Reason Codes for the IXLCACHE REQUEST=UNLOCK_CO_NAME Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=WRITE_DATA

Description

A WRITE_DATA request allows you to write data from the storage areas specified by BUFFER or BUFLIST and/or ADJAREA to a data entry in the cache structure. (A data entry is storage in the coupling facility cache structure comprised of one or more data elements where the system stores a data item. For more information, please see the *z/OS MVS Programming: Sysplex Services Guide*.) The NAME parameter identifies the data item that will be, or has already been, introduced to the structure. Once introduced to the structure, NAME also references the structure resources (the directory entry and data entry) allocated for that data item.

You can use the WRITE_DATA request to either:

- Create a new data entry to contain the data item. (A directory entry would also be allocated.)
- Update the contents of an existing data entry with the new or changed data item.

To register or update your interest in a data item, you must specify an index into your local cache vector (VECTORINDEX) to be associated with the data item in the structure. This index identifies a vector entry that cache services uses to indicate both your interest in the data item and the validity of your locally cached copy of the data item. Having registered interest in a data item **means** that you have a valid copy of the data item in your local cache buffer. When another user updates the data item in the structure, cache services will update the data item's associated vector entry, thereby deregistering your interest in the data item and invalidating your locally cached copy.

Cache services is responsible for deregistering your interest in a data item whenever another user updates the data item in the structure. You are responsible for maintaining the association between the entry in your local cache vector and the copy of the data item in your local cache buffer. You are also responsible for upholding any serialization protocols and procedures established to manage access to this shared data.

For structures allocated in a coupling facility of CFLEVEL=5 or higher, you can request that interest be deregistered in the entry specified by OLDNAME without registering interest in the entry specified by NAME.

The WHENREG parameter allows you to specify that the data be written only if you have already registered your interest in the data item: (To have a registered interest in the data item, the entry must already exist in the structure.)

- If you specify WHENREG=YES without VECTORINDEX, your interest must already be registered. (To have a registered interest in the data item, the entry must already exist in the structure and the vector index must be the same as that with which you registered interest.)
- If you specify WHENREG=YES with VECTORINDEX (valid only for structures in a CFLEVEL=2 and higher coupling facility), your interest must already be registered **and** the vector index specified must be the same as that with which you are currently registered. When VECTORINDEX is specified and the vector index does not equal the vector index with which you currently have a registered interest in the data item, the system fails the WRITE_DATA request and returns the vector index with which you are currently registered in the cache answer area.
- If you specify WHENREG=NO, the data item is written regardless of whether your interest is already registered. For a structure llocated in a coupling facility of CFLEVEL=4 or lower:
  - If the entry exists in the structure, your interest is updated.
  - If the entry does not exist in the structure, it is created and your interest is registered.

  VECTORINDEX is required when WHENREG=NO is specified for structures allocated in a coupling facility of CFLEVEL=4 or lower.

Optionally, for structures allocated in a coupling facility of CFLEVEL=5 or higher, you can use the REGUSER keyword to specify whether interest in the entry is to be registered or not.

- If REGUSER=YES is specified, interest in the entry is registered. When registration is performed, if connection interest is already registered, the specified VECTORINDEX replaces any previously specified local cache vector index for the entry.

  VECTORINDEX is required when REGUSER=YES is either specified or selected by default.
- If REGUSER=NO is specified, interest in the entry will not be registered. Be aware that entries in a data item for which no interest has been registered are higher-priority candidates for reclaim processing than those for which interest is registered.

  VECTORINDEX is required when REGUSER=NO is specified along with OLDNAME and is ignored if REGUSER=NO is specified without OLDNAME. When OLDNAME is also specified with VECTORINDEX, interest is deregistered in the entry specified by OLDNAME.

You can write either changed (CHANGED=YES) or unchanged (CHANGED=NO) data to the cache structure. If you write changed data (cache copy is different from permanent storage copy) you must assign the data item to a cast-out class (COCLASS). You must assign both changed and unchanged data items to a storage class (STGCLASS).

For structures allocated in a coupling facility with CFLEVEL=3 or lower, BUFFER or BUFLIST is required and ELEMNUM must be 1 or greater. For structures allocated in a coupling facility with CFLEVEL=4 or higher,

- When CHANGED=YES, BUFFER or BUFLIST is required and ELEMNUM must be 1 or greater.
- When CHANGED=NO, neither BUFFER nor BUFLIST is required and ELEMNUM can be 0 or greater.

You also have the option to invalidate other users' local copies of the data (CROSSINVAL) and to obtain the cast-out lock (GETCOLOCK), if necessary.

For structures allocated in a coupling facility of CFLEVEL=5 or higher, the VERSCOMP keyword allows you to specify a value that is to be compared with the version number associated with the entry being processed. The VERSCOMPTYPE keyword is used to define how the comparison is to be performed. If a version comparison mismatch occurs, processing terminates with no change to the structure. If a version comparison is successful, you can update (or choose not to update) the value of the entry version number by specifying the VERSUPDATE keyword.

For structures allocated in a coupling facility of CFLEVEL=9 or higher, the ASSIGN keyword on an IXLCACHE REQUEST=WRITE_DATA,WHENREG=NO request allows you to specify that the system is to suppress the creation of a new directory entry when an existing directory entry is not found.

For more information, see *z/OS MVS Programming: Sysplex Services Guide*.

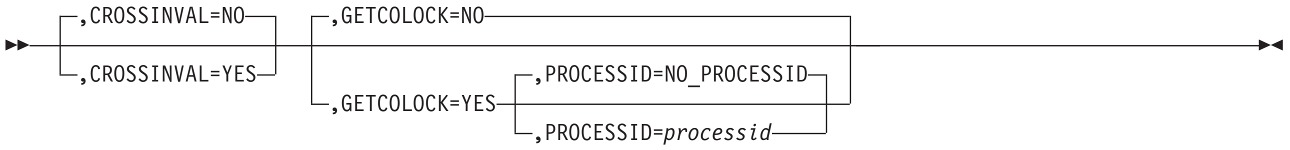
Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=WRITE_DATA is as follows:

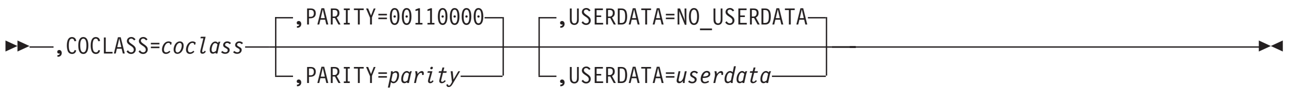
main diagram

```
                                             ┌─,VECTORINDEX=NO_VECTORINDEX─┐
                          ┌─,WHENREG=YES─────┴─,VECTORINDEX=vectorindex────┴──┐
►►──IXLCACHE──b──REQUEST=WRITE_DATA──┼──────────────────────────────────────────────┼─────────►
                          ├─,WHENREG=NO (for CFLEVEL<=4)─┤ parameters-5 ├─────┤
                          └─,WHENREG=NO (for CFLEVEL>=5)─┤ parameters-6 ├─────┘

  ┌─,VERSCOMP=NO_VERSCOMP──────────────────────────────┐  ┌─,VERSUPDATE=NONE──────────────────┐
►─┼────────────────────────────────────────────────────┼──┼───────────────────────────────────┼─►
  │                       ┌─,VERSCOMPTYPE=EQUAL───────┐ │  ├─,VERSUPDATE=INC───────────────────┤
  └─,VERSCOMP=verscomp────┼───────────────────────────┼─┘  ├─,VERSUPDATE=DEC───────────────────┤
                          └─,VERSCOMPTYPE=LESSOREQUAL─┘    └─,VERSUPDATE=SET─,NEWVERS=newvers──┘

  ┌─,CHANGED=NO─,CROSSINVAL=NO─,GETCOLOCK=NO─┐
►─┼──────────────────────────────────────────┼──,ELEMNUM=elemnum──,CONTOKEN=contoken────────►
  ├─,CHANGED=NO──┤ parameters-1 ├────────────┤
  └─,CHANGED=YES─┤ parameters-2 ├────────────┘

  ┌─,REQID=NO_REQID─┐                     (1)
►─┼─────────────────┼──,NAME=name───┬───────────────────────────────────────────────┬──────►
  └─,REQID=reqid────┘               ├─,BUFLIST=buflist─┤ parameters-3 ├─────────────┤
                                    └─,BUFFER=buffer─┤ parameters-4 ├─,BUFSIZE=bufsize─┘

  ┌─,ADJAREA=NO_ADJAREA─┐                       ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►─┼─────────────────────┼──,STGCLASS=stgclass───┼────────────────────────────────────────────┼─►
  └─,ADJAREA=adjarea────┘                       ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
                                                │                ┌─,REQDATA=NO_REQDATA─┐     │
                                                ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
                                                │                └─,REQDATA=reqdata────┘     │
                                                ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
                                                ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
                                                │                 ┌─,REQDATA=NO_REQDATA─┐    │
                                                ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
                                                │                 └─,REQDATA=reqdata────┘    │
                                                └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┘

  ┌─,ANSAREA=NO_ANSAREA───────────────┐
►─┼───────────────────────────────────┼──┬─────────────────────┬──┬─────────────────────┬──────►
  └─,ANSAREA=ansarea─,ANSLEN=anslen───┘  └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘

  ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┼───────────────────────────┼─────────────────────────────────────────────────────────────►
  ├─,PLISTVER=MAX─────────────┤
  └─,PLISTVER=plistver────────┘

   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼───────────────────────────────────────────────────►◄
   │                  ┌─,0D──────┐      │
   ├─,MF=(L─,mfctrl───┼──────────┼─)────┤
   │                  └─,mfattr──┘      │
   │                  ┌─,COMPLETE─┐     │
   └─,MF=(E─,mfctrl───┼───────────┼─)───┘
                      └─,COMPLETE─┘
```

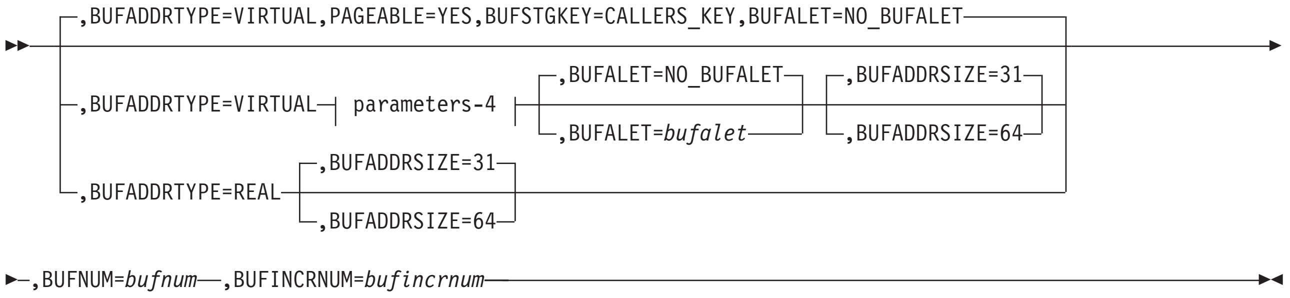
parameters-1



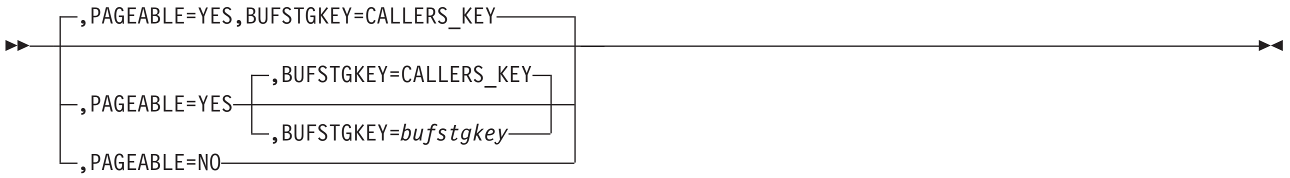
parameters-2



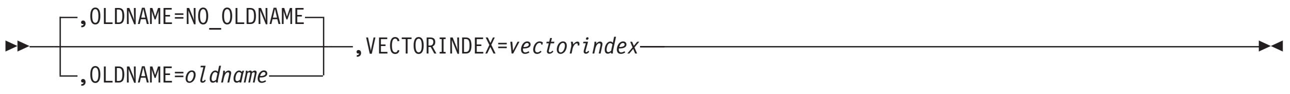
parameters-3



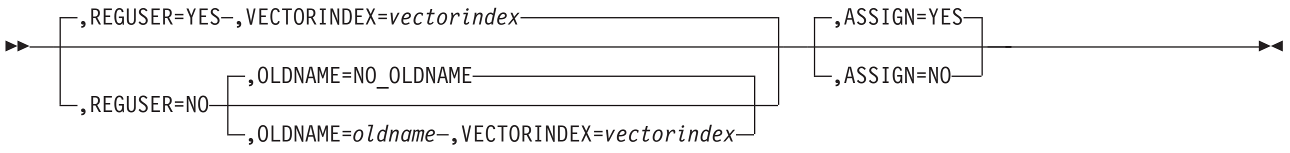
parameters-4



parameters-5



parameters-6



Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=WRITE_DATA are listed in alphabetical order. Default values are underlined:

REQUEST=WRITE_DATA

Use this input parameter to specify that the data item identified by NAME be written from the areas specified by BUFFER or BUFLIST, and/or ADJAREA to the cache structure.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*

Use this input parameter to specify a storage area to contain the adjunct data to be written to an entry in the cache structure. Specify ADJAREA only when ELEMNUM is greater than 0.

If the structure supports adjunct data and ADJAREA is not specified or ADJAREA=NO_ADJAREA is specified, binary zeros are written to the adjunct area. If the structure does not support adjunct data and ADJAREA is specified, ADJAREA is ignored. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area that contains the adjunct data you want associated with this data entry.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro. On a successful completion of a request, the following information is returned in the answer area:

- If changed data is written to the cache, the total number of entries containing either changed or locked-for-cast-out data, that are assigned to the storage class to which the entry was written, is returned (field CAATOTCHANGED).
- If changed data is written to the cache, the total number of data items assigned to the cast-out class to which data was just written is returned (field CAACOCOUNT).
- If WHENREG=NO was specified and the request replaced a previously specified vector index for the entry, the replaced vector index is returned (fields CAAINVLCVINUM and CAAINVLCVI).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request may be stored.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,ASSIGN=YES
,ASSIGN=NO

Use this input parameter to specify whether a directory entry will be assigned for NAME if one does not currently exist.

YES

A directory entry will be assigned.

NO

No directory entry will be assigned.

ASSIGN=NO is meaningful only for cache structures allocated in a coupling facility of CFLEVEL=9 or higher.

,BUFADDRSIZE=31

,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this input parameter to specify a buffer area to hold the entry data to be written to the cache structure.

You can define the buffer size to be a total of up to 65536 bytes. Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the data to be written to the entry in the structure.

,BUFINCRNUM=*bufincrnum*
Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*
Use this input parameter to specify a list of buffers to hold the entry data to be written to the cache structure. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 elements.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

The BUFLIST buffers must:

- All reside in the same address space or data space as defined by BUFALET.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes as defined by BUFINCRNUM.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. XES treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

For structures allocated in CFLEVEL=0 through CFLEVEL=3 coupling facilities, one of BUFFER or BUFLIST is required. For structures allocated in CFLEVEL=4 or higher coupling facilities, one of BUFFER or BUFLIST is required when CHANGED=YES and optional when CHANGED=NO.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte storage area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be written to the entry. If you do not want buffers, but want to specify this parameter, code BUFNUM=NO_BUFNUM.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CHANGED=NO
,CHANGED=YES

Use this input parameter to specify whether changed data is to be written to an entry in the cache structure.

NO

The data to be written is unchanged:
- The cached copy is the same as the permanent storage copy.
- In a store-through process, the permanent storage copy will be updated (by the user) to match the cached copy, either before the cast-out lock is released (if GETCOLOCK=YES was specified), or before serialization on the named data item is released.

If the entry already exists in the structure, the entry must contain unchanged data or the request will fail. You cannot overwrite changed data with unchanged data.

For structures allocated in CFLEVEL=4 or higher coupling facilities, ELEMNUM=0 and one of BUFFER and BUFLIST are optional when CHANGED=NO is specified.

YES

The data to be written is changed. The changed data will be assigned to the specified cast-out class (COCLASS) superseding any previously specified cast-out class for the data. With the exception of your connection, all users with registered interest in the data will have their interest deregistered such that their locally cached copies of the data are invalidated.

,COCLASS=*coclass*

Use this input parameter to specify the cast-out class to which the data item being written is to be assigned. Any previous assignment is updated to the new specification.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the cast-out class value.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CROSSINVAL=NO
,CROSSINVAL=YES

Use this input parameter to specify whether cross-invalidate processing should be performed when writing unchanged data.

NO

Cross-invalidate processing is not performed. All users with registered interest in the data item remain registered, and the copies of the data item in their local cache buffers remain valid.

YES

Cross-invalidate processing is performed. All users with registered interest in the data item, with the exception of your connection, have their interest deregistered and the copies of the data item in their local cache buffers invalidated.

,ELEMNUM=*elemnum*

Use this input parameter to specify the number of elements to be allocated to the data entry. Valid ELEMNUM values are from 0 to the MAXELEMNUM value that was returned to the connector in the connect answer area. For structures allocated in a CFLEVEL=0 coupling facility, ELEMNUM values can be in the range of 1 to 16. For structures allocated in coupling facilities with a CFLEVEL between

CFLEVEL=1 and CFLEVEL=3, ELEMNUM values can be in the range of 1 to 255. For structures allocated in coupling facilities with a CFLEVEL=4 or higher, ELEMNUM values can be in the range of 0 to 255 where 0 is valid only when CHANGED=NO.

Considerations for specifying ELEMNUM are:

- If this request creates a new entry, the number of elements is set to the ELEMNUM value.
- If the entry already exists, the number of elements is updated to this ELEMNUM value.

If the data passed in BUFFER/BUFLIST does not fill up all the space in the elements, the extra space will be padded with zeros. If the data passed in BUFFER/BUFLIST is greater than the amount of space in the elements, the data will be truncated.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of elements.

,GETCOLOCK=NO
,GETCOLOCK=YES

Use this input parameter to specify whether the cast-out lock should be obtained when writing unchanged data.

NO
The cast-out lock is not obtained.

YES
The cast-out lock is obtained.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,NAME=*name*
Use this input parameter to specify the name of the data item to be written to the cache structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the data item.

,NEWVERS=*newvers*
Use this input parameter to specify the value that is to be assigned to the entry version number.

NEWVERS is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the value for the entry version number.

,OLDNAME=NO_OLDNAME
,OLDNAME=*oldname*

Use this input parameter to specify the name of the data item for which your interest should be deregistered.

- For structures allocated in a coupling facility of CFLEVEL=4 or lower, you must also specify the name of the data item for which interest is to be registered after the interest is deregistered in OLDNAME. Identify the data item to which interest is to be registered with NAME. The VECTORINDEX currently associated with the entry specified by OLDNAME will be reassigned to the name of the data item specified by NAME.
- For structures allocated in a coupling facility of CFLEVEL=5 or higher, you can specify that interest in the name of the data item specified by OLDNAME is to be deregistered without registering interest in the entry specified by NAME.

In either case, whenever deregistration of interest is requested, VECTORINDEX must be specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the name of the old data item that was associated with VECTORINDEX.

,PARITY=00110000
,PARITY=*parity*

Use this input parameter to update the directory entry parity bits for the data item specified by NAME. The parity bits are updated only when CHANGED=YES is specified and the initial value of the directory entry parity bits is null (that is, B'11'). If these conditions are not met, the PARITY value is ignored.

Bits two and three of the 8-bit PARITY field are the parity bits. Valid parity values are:
- xx00xxxx - Parity equals 0
- xx01xxxx - Parity equals 1
- xx11xxxx - Null (parity is not set)

Note that when a directory entry is created, the parity bits are initialized to the null value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the parity bits.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,PROCESSID=NO_PROCESSID
,PROCESSID=*processid*

Use this input parameter to specify a user-defined process identifier to be compared with the cast-out lock along with the connection identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the user-defined process identifier.

,REGUSER=YES
,REGUSER=NO

Use this input parameter to specify, for structures allocated in coupling facilities with CFLEVEL=5 or higher, whether the request should register connection interest in the entry.

YES

Specify this option to indicate that connection interest registration will be performed.

VECTORINDEX is required when REGUSER=YES is either specified or selected by default.

NO

Specify this option to indicate that no connection interest will be registered. Be careful when using REGUSER=NO because preference to reclaiming is given to entries that have data but no registered interest.

VECTORINDEX is required when REGUSER=NO is specified along with OLDNAME for the deregistration of interest. VECTORINDEX is ignored when REGUSER=NO is specified without the specification of OLDNAME.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STGCLASS=*stgclass*
Use this input parameter to assign a storage class to the data item being written. Any previous assignment is updated to the new specification.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage class.

,USERDATA=NO_USERDATA
,USERDATA=*userdata*
Use this input parameter to specify user-defined information to be written to the directory entry for the data item specified by NAME. The information is written only if either of the following is true:

- There is no entry data in the structure for NAME.
- There is unchanged entry data in the structure for NAME.

If one of these conditions is not met, USERDATA is ignored. If you do not want to write user data, but want to specify this parameter, code USERDATA=NO_USERDATA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user data.

,VECTORINDEX=*vectorindex*

Use this input parameter to specify an index into your local cache vector to be associated with the data item specified by NAME. The vector index identifies a vector entry that cache services will use to indicate both your interest in the data item and the validity of the copy of the data item in your local cache buffer.

- With WHENREG=NO, for structures allocated in a coupling faciltiy with CFLEVEL=4 or lower, VECTORINDEX is required. If the vector index you specify is already associated with another data item, you must disassociate the vector index from the old name by specifying OLDNAME before the vector index can be associated with the data item specified by NAME.

  For structures allocated in a coupling facility with CFLEVEL=5 or higher, VECTORINDEX is required when either REGUSER=YES or OLDNAME is specified with WHENREG=NO.
- With WHENREG=YES, which is valid only when the cache structure is allocated in a CFLEVEL=2 or higher coupling facility, the vector index you specify must be the same as the vector index with which you currently have a registered interest in the data item. When VECTORINDEX is specified and the vector index does not equal the vector index with which you currently have a registered interest in the data item, the request fails. The vector index with which you are currently registered is returned in the cache answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the vector index for NAME.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of the entry designated by NAME.

VERSCOMP is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

If the condition specified by VERSCOMPTYPE is not met, the request is terminated with no resultant change to the structure.

Note that use of VERSCOMP is needed to ensure that updates to the version number requested with the VERSUPDATE keyword are not processed multiple times as a result of internal request redrive logic. When VERSCOMP is requested along with VERSUPDATE to update the version number, then if the initial execution of the request succeeds, any subsequent internal redrive of the request will fail due to a version number miscompare, preventing multiple updates from occurring on the request. Conversely, if the initial execution of the request was unsuccessful, then any subsequent internal redrive of the request will be able to execute successfully and update the version number only once.

In either of these cases, if the request is internally redriven and experiences a version number miscompare on the redrive, a return and reason code of IXLRSNCODESTATUSUNKNOWN will be returned. This reflects the fact that it is not known whether the observed version number miscompare resulted from the version number update succeeding on the original issuance of the request (causing the miscompare on the redriven request), or the observed version number miscompare was present all along, or resulted from a version number update made by another request. When this return and reason code is returned, it is up to the user to determine whether or not the requested update has actually occurred, and take the appropriate recovery action.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the comparative version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the structure entry version number comparison is to be performed.

VERSCOMPTYPE is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

VERSCOMPTYPE=EQUAL
The version number for the structure entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the structure entry must be less than or equal to the value specified for VERSCOMP.

,VERSUPDATE=NONE
,VERSUPDATE=INC
,VERSUPDATE=DEC
,VERSUPDATE=SET

Use this input parameter to specify how the entry version number will be updated or, for those cases where an entry is created, initialized.

VERSUPDATE is meaningful only for structures allocated in a coupling facility of CFLEVEL=5 or higher.

Note that, depending on the VERSUPDATE specification, internal request redrive logic may cause a request to INCrement the version number more than once, DECrement the version number more than once, or SET the version number more than once. If it is necessary to ensure that this kind of multiple operation does not occur, version number comparison must also be requested. See the description of the VERSCOMP keyword.

VERSUPDATE=NONE
The version number is not updated.

On a request that causes an entry to be created, the version number is set to contain all binary zeros.

VERSUPDATE=INC
The version number will be incremented.

On a request that causes an entry to be created, the version number for the created entry is set to contain all binary zeros except for the low order bit, which is set to one.

VERSUPDATE=DEC
The version number will be decremented.

On a request that causes an entry to be created, the version number for the created number is set to contain binary ones.

VERSUPDATE=SET
The version number will be set to the value specified by NEWVERS, including the case where an entry is created.

,WHENREG=YES
,WHENREG=NO

Use this input parameter to specify whether previous registration of interest by the requesting user in the data item specified by NAME is a prerequisite to this request.

YES
The request is processed only if your interest in the data item is already registered. If your interest is not registered, the request will fail.

NO
The request is processed regardless of whether your interest in the data item is already registered.

- If the entry already exists in the structure, the data and your interest is updated.
- If the entry does not exist in the structure, a new entry is created, and your interest is registered.

See the VECTORINDEX parameter description for how to register interest in a data item.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The VECTORINDEX specified is not valid. Request processing was suppressed. **Action:** Specify a vector index that is within the current range of the number of vector entries for the local vector requested for this structure. The number of vector entries is determined by the VECTORLEN parameter specified on the IXLCONN macro and may be changed by issuing the IXLVECTR macro.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. • The request with WHENREG=YES specified failed for one of the following reasons: – The entry designated by name is not in the cache structure. – The user's connection did not have registered interest in the entry. – With CFLEVEL=2, if VECTORINDEX was specified, the vector index did not match the vector index with which the user currently has a registered interest in the data item. The currently-registered vector index is returned in the cache answer area (field CAALCVINUM, which is valid only if CAALCVI is set on). • The request with WHENREG=NO,ASSIGN=NO specified resulted in the suppression of the creation of a new data entry because the entry did not previously exist. (CFLEVEL=9 or higher) **Action:** If this is expected, then your copy of the entry is invalid and you must get the latest copy. Re-read the data item into your local cache buffer and apply your changes again. Then try the write again. If this is unexpected, try the following: • Verify that NAME is uncorrupted. • Verify that the CONTOKEN is for the correct structure. • Verify that the VECTORINDEX is the same as that for which you have previously registered interest in the entry. You might want to rerun the request with WHENREG=NO to allocate a directory entry for this NAME.
8	xxxx0826	**Equate Symbol:** IXLRSNCODEINCOMPATSTATE **Meaning:** Program error. The request failed because the state of the named data item is incompatible with the request. The following fields are returned in the answer area: • CAACHANGED • CAAINVLCVINUM (if WHENREG=NO was specified) • CAACOLOCKSTATE • CAACOLOCKVAL **Action:** Check for the following conditions: • GETCOLOCK was specified for an entry that contains changed data (see CAACHANGED, CAACOLOCKSTATE, and CAACOLOCKVAL to determine if this is the case). • WHENREG=NO was specified and there is registered interest in the data item specified by NAME (see CAAINVLCVINUM).
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified by STGCLASS exceeds the maximum number of storage classes defined for the cache structure. **Action:** The number of storage classes allowed for a structure are defined on the IXLCONN macro by the NUMSTGCLASS parameter. Correct the STGCLASS parameter to specify a valid storage class.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx082E	**Equate Symbol:** IXLRSNCODEBADCOCLASS **Meaning:** Program error. The cast-out class specified by COCLASS exceeds the maximum number of cast-out classes defined for the cache structure. **Action:** Correct the COCLASS parameter to specify a valid cast-out class. The valid range for COCLASS is from 1 to the maximum value specified by the NUMCOCLASS parameter on the IXLCONN macro.
8	xxxx082F	**Equate Symbol:** IXLRSNCODEBADPARITY **Meaning:** Program error. The parity value specified by PARITY is not valid. **Action:** The value specified in the request contained parity bits that were not valid. The value for the parity bits may be 00, 01, or 11 in bits 2 and 3 of the PARITY specification (xxpp xxxx where pp is the parity bits). See the explanation for the PARITY keyword in this section.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in a BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the real buffer addresses specified by BUFLIST are valid.
8	xxxx0837	**Equate Symbol:** IXLRSNCODEBADWRITEADJDATA **Meaning:** Program error. The request specified that adjunct data was to be written, but the area specified by ADJAREA is not addressable. There was no change to the structure. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** Program error. The specified structure entry has a version number which does not meet the criteria specified in the request. **Action:** None necessary.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the caller's PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information).
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The value specified for ELEMNUM is not valid. **Action:** Check the element information specified on the IXLCONN for this structure. The number of elements is affected by MAXELEMNUM on IXLCONN.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.

Table 38. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATA Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. Allocation of a directory entry and/or data entry was necessary, but was unavailable or could not be reclaimed. The answer area field, CAASTGCLFULL, contains the storage class from which the reclaiming operation failed. **Action:** • Determine if any data items may be cast-out to make room for this item. • Check your usage of storage classes to see if some data items can be moved to a different storage class (preferably with a lower priority) so some entries in the structure can be freed. • Determine if a rebuild of the structure is necessary to make room for more data entries/items.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCACHE REQUEST=WRITE_DATALIST

Description

A WRITE_DATALIST request allows you to write data to a cache structure for a given set of entries for a connection specified by CONTOKEN. Each entry to be written to the structure is identified by a write-operation-block (a WOB, mapped by IXLYWOB) contained in the BUFFER or BUFLIST storage area. When BUFFER is specified, 1 to 256 write-operation-blocks are allowed. When BUFLIST is specified, 1 to 16 write-operation-blocks are allowed.

The system references the write-operation-blocks by an index into BUFFER or BUFLIST and an offset specifying the location of the data area to be processed. The entries are processed sequentially beginning with the write-operation-block specified by the first index entry (STARTINDEX) and ending with the write-operation-block specified by the last index entry (ENDINDEX). The write-operation-block entries are numbered starting with 1. DATAOFFSET contains the offset in 256-byte increments into the storage area pointed to bu BUFFER or BUFLIST. This is where the data resides that coincides with the write-operation-blocks.

Each write-operation-block contains the following information:
- Name of the entry
- Data area size in terms of the number of elements
- Storage class
- Additional information needed to write an entry to the cache structure

As each write-operation-block is processed, information about the request is returned in a write-operation-response-block (WORB). There is a one-to-one correspondence between each WOB in the BUFFER or BUFLIST and a WORB that the system returns in WORBAREA. The contents of each WORB, mapped by IXLYWORB, are as follows:
- The castout count for the castout class to which data was just written
- Total changed count for the storage class to which data was just written
- The invalidated local cache vector index, if any
- Additional information related to writing the data to the cache structure.

Additionally, the answer area (ANSAREA) contains the invalidated local cache validity vector, which is updated for each WOB processed.

When you issue a WRITE_DATALIST request, based on the contents of the WOB, you can specify whether you do not want to register interest in the entry, whether you want to deregister any outstanding interest for a different entry, whether a directory entry is to be assigned, and, depending on the value of a change control indicator in the WOB, how registered interest and cross-invalidations are to be performed.
- The suppress registration indicator specifies whether the request should register connection interest in the entry.
- The assignment suppression control indicator specifies whether a directory entry should be assigned if one does not currently exist.
- OLDNAME specifies the name of the data item for which your interest should be deregistered. If both OLDNAME and NAME are specified in a WOB and their values are not equal, the name replacement control indicator specifies whether deregistration of interest for OLDNAME should be performed.
- The change control indicator specifies whether changed data is to be written to an entry in the cache structure.

See *z/OS MVS Programming: Sysplex Services Guide* for a complete list of WOB indicators and how their settings affect the processing of each entry.

Processing for a WRITE_DATALIST request can be discontinued in the following instances:
- Version number comparison fails.
- Cast-out class specified is not valid.
- Parity bits specified are not valid.
- Inconsistency in the changed status of the data.
- Data area size specified is not valid.
- Storage class specified is not valid.
- Inability to obtain resources from the storage class specified.
- Inaccurate element number specified.
- Entry does not exist.

In these cases, all of the prior write-operation-blocks were processed and the index of the failing write-operation-block is returned in the answer area (CAAWDLINDEX). See *z/OS MVS Programming: Sysplex Services Guide* for a complete description of the instances listed above.

Processing of an entire WRITE_DATALIST request can be suppressed in the following instances:
- The write-operation-block contains both the change control indicator and the get cast-out lock control indicator set.
- The write-operation-block contains a local cache vector index that is not valid.

In these cases, none of the specified write-operation-blocks was processed and the index of the failing write-operation-block is returned in the answer area (CAAWDLINDEX).

A WRITE_DATALIST request can complete prematurely because the request exceeds the time-out criteria for the coupling facility (time-out criteria is model-dependent) or because the WORBAREA becomes full before processing is complete for the requested WOBs. When a request completes prematurely, the system returns an index value (CAAWDLINDEX) in the answer area that you can use to restart the WRITE_DATALIST request. The index value when a WRITE_DATALIST completes prematurely is the index of the first unprocessed write-operation-block. Also returned in the answer area is the offset in the data block of the data area for the next write-operation-block to be processed and the invalidated local cache validity vector. All prior write-operation-blocks were processed.

To continue processing the remaining write-operation-blocks, reissue the WRITE_DATALIST request specifying the write-operation-block index and the offset of the next data area returned in the answer area as the STARTINDEX and the DATAOFFSET.

A WRITE_DATALIST request can be issued only for cache structures allocated in a coupling facility of CFLEVEL=12 or higher. WRITE_DATALIST requests issued for a cache structure allocated in a coupling facility of CFLEVEL=0 through 11 will fail.

For more information, see *z/OS MVS Programming: Sysplex Services Guide*.

Syntax Diagram

The syntax diagram for IXLCACHE REQUEST=WRITE_DATALIST is as follows:

main diagram

```
►►──IXLCACHE──b──REQUEST=WRITE_DATALIST──,STARTINDEX=startindex──,ENDINDEX=endindex──────────────►

                                                           ┌─,REQID=NO_REQID─┐
►─,DATAOFFSET=dataoffset──,WORBAREA=worbarea──,CONTOKEN=contoken─┴─────────────────┴──────────────►
                                                           └─,REQID=reqid────┘

                                                        ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►──┬─,BUFLIST=buflist─┤ parameters-3 ├──────────────────┬─┼────────────────────────────────────────────┼──►
   └─,BUFFER=buffer─┤ parameters-4 ├─,BUFSIZE=bufsize─┘   ├─,MODE=SYNCECB─,REQECB=reqecb─────────────────┤
                                                        │                 ┌─,REQDATA=NO_REQDATA─┐      │
                                                        ├─,MODE=SYNCEXIT──┼─────────────────────┼──────┤
                                                        │                 └─,REQDATA=reqdata────┘      │
                                                        ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken───────────┤
                                                        ├─,MODE=ASYNCECB─,REQECB=reqecb────────────────┤
                                                        │                  ┌─,REQDATA=NO_REQDATA─┐     │
                                                        ├─,MODE=ASYNCEXIT──┼─────────────────────┼─────┤
                                                        │                  └─,REQDATA=reqdata────┘     │
                                                        └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken──────────┘

   ┌─,ANSAREA=NO_ANSAREA───────────────┐
►──┼───────────────────────────────────┼──┬────────────────────┬──┬────────────────────┬──────────────►
   └─,ANSAREA=ansarea─,ANSLEN=anslen───┘  └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S────────────────────────────────┐
►──┼───────────────────────────┼───┼──────────────────────────────────────┼──────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤   │                   ┌─,0D──────┐       │
   └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl────┼──────────┼─)─────┤
                                   │                   └─,mfattr──┘       │
                                   │                   ┌─,COMPLETE─┐      │
                                   └─,MF=(E─,mfctrl────┼───────────┼─)────┘
                                                       └─,COMPLETE─┘
```

parameters-3

```
    ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET──────────────────┐
►►──┼─────────────────────────────────────────────────────────────────────────────────────────────┼────►
    │                                                ┌─,BUFALET=NO_BUFALET─┐ ┌─,BUFADDRSIZE=31─┐  │
    └─,BUFADDRTYPE=VIRTUAL─┤ parameters-4 ├──────────┼─────────────────────┼─┼─────────────────┼──┘
                                                     └─,BUFALET=bufalet────┘ └─,BUFADDRSIZE=64─┘

►─,BUFNUM=bufnum──────────────────────────────────────────────────────────────────────────────────►◄
```

parameters-4

```
    ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────────┐
►►──┼──────────────────────────────────────────────┼───────────────────────────────────────────────►◄
    │                 ┌─,BUFSTGKEY=CALLERS_KEY─┐   │
    ├─,PAGEABLE=YES───┼────────────────────────┼───┤
    │                 └─,BUFSTGKEY=bufstgkey───┘   │
    └─,PAGEABLE=NO─────────────────────────────────┘
```

Note: If you specify MODE=SYNCTOKEN or MODE=ASYNCTOKEN, then you must also specify ANSAREA=*ansarea*,ANSLEN=*anslen*.

Parameter Descriptions

The parameter descriptions for REQUEST=WRITE_DATALIST are listed in alphabetical order. Default values are underlined:

REQUEST=WRITE_DATALIST
Use this input parameter to specify that data for a given set of entries for a connection specified by CONTOKEN is to be written to the cache structure. Each entry to be written to the cache structure is identified by a write-operation-block, mapped by IXLYWOB.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by the IXLYCAA mapping macro. On a successful completion of a request, the following information is returned in the answer area:

- If changed data is written to the cache, the total number of entries containing either changed or locked-for-cast-out data, that are assigned to the storage class to which the entry was written, is returned (field CAATOTCHANGED).
- If changed data is written to the cache, the total number of data items assigned to the cast-out class to which data was just written is returned (field CAACOCOUNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request may be stored.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Use either CAALEVEL0LEN or CAALEVEL1LEN of the IXLYCAA mapping macro to determine the minimum size of the answer area. The answer area length must be at least large enough to accomodate the level of the IXLYCAA mapping appropriate to the requested function. When the value of PLISTVER is 0 — 3, the minimum answer area length is CAALEVEL0LEN; when the value of PLISTVER is 4 — 6, the minimum answer area length is CAALEVEL1LEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the length of the answer area (ANSAREA).

,BUFADDRSIZE=<u>31</u>
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=<u>VIRTUAL</u>
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

Note that for WRITE_DATALIST requests, real storage addresses cannot be used.

,BUFALET=<u>NO_BUFALET</u>

,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to hold the entry data to be written to the cache structure.

You can define the buffer size to be a total of up to 65536 bytes. Other requirements depend on the size you select:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) to contain the data to be written to the entry in the structure.

,BUFINCRNUM=16

For WRITE_DATALIST requests that specify a BUFLIST parameter, the system assumes a BUFINCRNUM value of 16 that corresponds to a buffer size of 4096.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold information about the entry data to be written to the cache structure. BUFLIST specifies a 128-byte storage area that consists of a list of 1 to 16 elements.

The format of the list is a set of 8-byte elements. The BUFADDRSIZE keyword denotes whether four or eight bytes of the element are used.

- If BUFADDRSIZE=31 is specified, then the first four bytes of each element are reserved space and the last four bytes contain the address of the buffer.
- If BUFADDRSIZE=64 is specified, then the full eight bytes specify the address of the buffer.

For WRITE_DATALIST requests, the first buffer pointed to by the first element in the BUFLIST can contain up to 16 WOBs; the remainder of the buffers in BUFLIST can contain data to be written to the entries specified by the WOBs. The first buffer must reside in 31-bit virtual storage. All other buffers pointed to by BUFLIST can reside in either 31-bit or 64-bit virtual storage. Real storage addresses cannot be used for any buffers in BUFLIST.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte storage area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
: Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be written to the entry. If you do not want buffers, but want to specify this parameter, code BUFNUM=NO_BUFNUM.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the buffer list.

,BUFSIZE=*bufsize*
: Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
: Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

  If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
: Use this input parameter to specify the connect token that was returned by the IXLCONN service in the IXLCONN answer area, mapped by IXLYCONA. The connect token uniquely identifies your connection to the cache structure, and must be specified on each IXLCACHE invocation.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DATAOFFSET=*dataoffset*
: Use this input parameter to specify the offset in 256-byte increments into the data block in the storage area specified by BUFFER or BUFLIST of the first data area to be processed (the data block corresponding to the write-operation-block located by STARTINDEX).

  A value of 0 implies that there is no actual data being passed in the storage area specified by BUFFER or BUFLIST. When this occurs, all WOBs have to specify an ELEMNUM of 0 to indicate that no data is being passed in to go with those WOBs. If any WOB specifies an ELEMNUM other than 0, the request will fail with return code IXLRETCODEPARMERROR, reason code IXLRSNCODEBADELEMNUM.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the offset into the data block.

,ENDINDEX=*endindex*
: Use this input parameter to specify the ending index for the last WOB to be processed in the storage area specified by BUFFER or BUFLIST. The index value must be greater than or equal to the value specified for STARTINDEX, but less than or equal to 256.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the ending index value.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**

,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND

The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCACHE Version Support" on page 248 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*
Use this input parameter to specify the index for the first WOB in the storage area specified by BUFFER or BUFLIST to be processed. Valid STARTINDEX values are from 1 to the value of ENDINDEX. The first WOB in the storage area pointed to by the BUFFER or BUFLIST parameter has index number 1.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field containing the starting index value.

,WORBAREA=*worbarea*
Use this output parameter to contain the write-operation response block array.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 192-byte area to contain the write-operation response block data.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYCAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, or ASYNCTOKEN, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If you specified MODE=SYNCECB, the ECB specified in REQECB will be posted when the request has finished. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has finished. • If you specified MODE=SYNCTOKEN, the REQTOKEN contains an asynchronous request token that may be used on the IXLFCOMP macro to determine when the request has completed. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's completion exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because the model-dependent time-out criteria of the coupling facility has been exceeded. The index of the first unprocessed write-operation-block and the offset of the current data area have been returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks have been processed. **Action:** To restart the request, update STARTINDEX with the value of CAAWDLINDEX and DATAOFFSET with the value of CAAWDLDATAOFFSET. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The parameter list (mfctrl) for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** • Verify the parameter list address. • The parameter list must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are issuing this macro while disabled, the parameter list must reside in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before this macro.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running. • Verify that your program is running on an MVS system that supports the version of the macro you are using.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure. The numbers on the following actions correspond to the numbers above. 1. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know of the disconnection. 2. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure to determine why you did not know that the task ended. 3. Issue IXLCONN to connect to the structure. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued. 5. Participate in the rebuild. When it is complete, try again. 6. Discontinue use. Perform recovery and cleanup for the structure and check protocol for use of structure. You may want to issue IXCQUERY to get more information about the structure.
8	xxxx0819	**Equate Symbol:** IXLRSNCODEBADVECTOROP **Meaning:** The vector index in the write-operation-block indexed by CAAWDLINDEX is not valid. None of the write-operation-blocks have been processed. **Action:** Correct the vector index value and reissue the WRITE_DATALIST request with the same STARTINDEX and ENDINDEX values.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** The connect token specified by CONTOKEN is not to a cache structure. **Action:** Verify the connect token for this cache structure. **Note:** The connect token can be found in the IXLCONN answer area. Verify the value specified on the TYPE= parameter of the IXLCONN request for this structure.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The request failed because the assignment suppression control indication specified in the write-operation-block was set and the entry did not exist. The data is not written. The index of the failing write-operation-block and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:** Reissue the request specifying for STARTINDEX the index of the next valid name element to be processed and DATAOFFSET with the offset of the next data area to be processed.
8	xxxx0826	**Equate Symbol:** IXLRSNCODEINCOMPATSTATE **Meaning:** Program error. The request failed because the state of the named data item is incompatible with the request. This may be due to one of the following conditions: • The change control indicator was not set in the write-operation-block, but the data in the cache structure was marked as changed. • The get cast-out lock control was set in the write-operation-block, but the cast-out lock was already held from a previous request. • The get cast-out lock control was set, but the entry was marked as changed or is already locked for castout by a CASTOUT_DATA or CASTOUT_DATALIST request. • The get cast-out lock control was set, but the entry is already locked for castout by another connector. • The change control was not set, but the entry is marked as changed. • The change control was not set, but the entry is locked for castout by a CASTOUT_DATA or CASTOUT_DATALIST request. The data is not written. The index of the failing write-operation-block (CAAWDLINDEX), the offset in the data block of the data area for the write-operation-block being processed (CAAWDLDATAOFFSET), the changed indication (CAACHANGED), the cast-out lock value (CAACOLOCKVAL), the cast-out lock state (CAACOLOCKSTATE), and the value of the local cache entry number (CAALCVINUM) are returned in the answer area. All prior write-operation-blocks were processed. **Action:** Check the accuracy of the following values: CAAWDLINDEX, CAAWDLDATAOFFSET, CAACHANGED, CAACOLOCKVAL, CAACOLOCKSTATE, and CAALCVINUM. Correct the values as necessary and resubmit the request.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx082D	**Equate Symbol:** IXLRSNCODEBADSTGCLASS **Meaning:** Program error. The storage class specified in the write-operation-block exceeds the maximum defined storage class for the structure. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:** Correct the storage class value in the write-operation-block indexed by CAAWDLINDEX and reissue the WRITE_DATALIST request starting with that write-operation-block.
8	xxxx082E	**Equate Symbol:** IXLRSNCODEBADCOCLASS **Meaning:** Program error. The cast-out class specified in the write-operation-block exceeds the maximum number of cast-out classes defined for the cache structure. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:** Correct the COCLASS value in the write-operation-block to specify a valid cast-out class. The valid range for COCLASS is from 1 to the maximum value specified by the NUMCOCLASS parameter on the IXLCONN macro.
8	xxxx082F	**Equate Symbol:** IXLRSNCODEBADPARITY **Meaning:** Program error. The parity value specified in the write-operation-block was not valid. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:** The value specified in the request contained parity bits that were not valid. The value for the parity bits may be 00, 01, or 11 in bits 2 and 3 of the PARITY specification (xxpp xxxx where pp is the parity bits).
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES) but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being nonpageable (PAGEABLE=NO) but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area(s) were not previously freed. • If BUFLIST was specified and your program is running in AR mode, ensure that: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used for BUFFER or for a buffer within the BUFLIST. • The buffer area(s) were not previously freed. • The buffer area(s) were allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If the caller is running in AR-mode and the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is running in AR-mode, SYSSTATE ASCENV=AR must be specified before issuing this macro. • If BUFLIST was specified and your program is running in AR mode the BUFALET specification is correct.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** • Verify the ANSAREA address. • ANSAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLCACHE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the ANSAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that the request token area specified by REQTOKEN is valid: • Verify the REQTOKEN address. • If you are calling IXLCACHE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLCACHE in AR-mode and you specified the REQTOKEN address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLCACHE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCACHE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area provided for the request, and rerun your program. Depending on the macro version number, specify the length of the answer area as follows: • CAALEVEL0LEN indicates the level 0 mapping of the answer area. • CAALEVEL1LEN indicates the level 1 mapping of the answer area. IXLCACHE macro invocations at the version 4 and higher level require an answer area length of CAALEVEL1LEN.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** Program error. The version number specified in the write-operation-block does not meet the version number criteria specified in the write-operation-block. The data is not written. The index of the write-operation-block that failed, the offset in the data block of the data area for the write-operation-block being processed, and the structure entry version number are returned in the answer area (fields CAAWDLINDEX, CAAWDLDATAOFFSET, and CAAVERSION). All prior write-operation-blocks were processed. **Action:**
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value or become enabled (release the CPU lock); then reissue the request.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0865	**Equate Symbol**: IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the caller's PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information).
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that: • The correct BUFLIST address was used. • The BUFLIST area was not previously freed. • If the caller is running in AR-mode and the BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If the caller is disabled, then the BUFLIST must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The ELEMNUM specified in the write-operation-block is not valid. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being porcessed are returned in the answer area (fields CAAWDLINDEX and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed. **Action:**

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0874	**Equate Symbol:** IXLRSNCODEBADWDLINDEX **Meaning:** Program error. The value specified for either STARTINDEX or ENDINDEX is not valid. No entries are processed. **Action:** Ensure that: • The values specified by STARTINDEX and ENDINDEX are in the range of 1 to 256. • The value specified by ENDINDEX is greater than or equal to the value of STARTINDEX. • The value specified by ENDINDEX does not imply a larger BUFFER size than was actually specified on the WRITE_DATALIST request.
8	xxxx087D	**Equate Symbol:** IXLRSNCODEBADWORBAREA **Meaning:** Program error. The storage area specified by WORBAREA is not addressable. **Action:** Ensure that: • The address passed in WORBAREA is valid. • The WORBAREA area was not previously freed. • The WORBAREA area is addressable from the caller’s primary address space or from the caller’s PASM access list. • If the caller is running in AR-mode and WORBAREA was specified using explicit register notation, ensure that the corresponding access register was updated appropriately. • If the caller is disabled, then WORBAREA must reside in either nonpageable or disabled reference storage. • If your program is running in AR mode, ensure that you specified SYSSTATE ASCENV=AR before issuing the IXLCACHE macro.
8	xxxx08AA	**Equate Symbol:** IXLRSNCODEELEMNUMMISMATCH **Meaning:** Program error. The specified data area size in the write-operation-block does not match the actual size of the corresponding data area in the data block. The data is not written. The index of the write-operation-block that failed and the offset in the data block of the data area for the write-operation-block being processed are returned in the answer area (fields CAAWDLINDEX, and CAAWDLDATAOFFSET). All prior write-operation-blocks were processed.) **Action:**
8	xxxx08AB	**Equate Symbol:** IXLRSNCODEBADDATAOFFSET **Meaning:** Program error. A DATAOFFSET was specified that is not valid. No entries are processed and no data is returned. **Action:**None required.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx08AC	**Equate Symbol:** IXLRSNCODEBADGETCOLOCK **Meaning:** Program error. In the write-operation-block the change control indicator was set and the get castout lock control indicator was also set. The data is not written. The cast-out lock is not obtained and the index of the failing write-operation-block is returned in the answer area (field CAAWDLINDEX). None of the specified write-operation-blocks were processed. Processing of the entire request was suppressed.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to the cache structure exists. This may occur due to operator commands such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as facility or path failures. The CONTOKEN is invalidated. The request fails. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD).
C	xxxx0C13	**Equate Symbol**: IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Quiesce, Rebuild Stop, or Rebuild Cleanup event. • The connector invoked IXLREBLD REQUEST=COMPLETE. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The IXLCACHE request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** Verify the validity of your data by comparing the expected results with what is in the coupling facility.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. Allocation of a directory entry was necessary, but was unavailable or could not be reclaimed. The data was not written. In the answer area, CAASTGCLFULL contains the storage class from which the reclaiming operation failed, CAAWDLINDEX contains the index of the write-operation-block that was being processed when the error occurred, and CAAWDLDATAOFFSET contains the offset in the data block of the data area for the write-operation-block being processed. All prior write-operation-blocks were processed. **Action:** • Determine if any data items may be cast-out to make room for this item. • Check your usage of storage classes to see if some data items can be moved to a different storage class (preferably with a lower priority) so some entries in the structure can be freed. • Determine if a rebuild or an alter of the structure is necessary to make room for more data entries/items. After correcting the error, restart the WRITE_DATALIST request starting with the write-operation-block indexed by CAAWDLINDEX.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The cache structure failed prior to completion of the request. **Action:** Attempt to rebuild the structure using IXLREBLD, or disconnect from the structure using IXLDISC.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Disconnect from the structure (using IXLDISC) and request that the installation provide a coupling facility of the correct CFLEVEL (CFLEVEL=12 or higher).
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. The coupling facility hardware might not be present. **Action:** XES may not be used when XCF is running local mode. XES requires that the coupling facility be installed.

Table 39. Return and Reason Codes for the IXLCACHE REQUEST=WRITE_DATALIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
10	xxxx10xx	**Equate Symbol:** IXLRSNCODECOMPERROR **Meaning:** System error. XES processing failure. The state of the structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Save the reason code information, and contact the IBM support center.

IXLCONN — Connect to a Coupling Facility Structure

Description

IXLCONN allows you to allocate and connect to a structure in the coupling facility or to connect to an already allocated structure. The first user to allocate the structure defines the structure attributes. Subsequent users can connect to the allocated structure but cannot change the structure attributes established when the structure was allocated. All connectors, whether the first or subsequent, are informed of the structure attributes through the connect answer area, mapped by IXLYCONA. It is the connector's responsibility to check the structure attributes to verify their acceptability.

- **Providing Structure Information**

  Structure attributes include a structure name (STRNAME), a structure disposition (STRDISP), and a structure type (TYPE).
  - You identify the structure to which you want to connect by name (STRNAME). For connections to be successful, structure names must be defined in the active CFRM policy, and the system on which you are running must be able to access the CFRM couple data set that contains the active CFRM policy. The system on which you are running also must have connectivity to a coupling facility that is in the preference list of the structure, as defined in the active CFRM policy.
  - STRDISP determines whether the structure remains allocated when all users have disconnected from the structure.
  - TYPE indicates the kind of structure — a cache, list, or lock structure. You can specify one of these types on an IXLCONN request. A set of attributes specific to each structure type allows you to define structure characteristics (for example, the size of data entries for a cache or list, or the number of lock users).
- **Providing Connection Information**

  You also must specify certain information about the connection on IXLCONN, such as the connection disposition (CONDISP), an event exit (EVENTEXIT), and an answer area (ANSAREA) and length (ANSLEN).
  - CONDISP determines whether a connection can enter a failed-persistent state which, in the event of a connection failure, would allow you to reconnect to the structure, if possible. Users can specify a connect name (CONNAME) on IXLCONN that they can use to reconnect to the structure if they fail, enter a failed-persistent state, and are subsequently restarted.
  - EVENTEXIT specifies the name of the event exit. The event exit gets control whenever an event about the structure or a connection to the structure (such as a failure of the connection) occurs. The system reports an event to the event exit of each connected user. Users establish protocols to process these events.
  - IXLCONN returns information about the structure and the connection in an area specified by ANSAREA. Included in this area is a connect token (CONTOKEN) that uniquely identifies the connection within the sysplex. After you have connected to a structure, you specify the CONTOKEN returned from IXLCONN on structure requests like IXLCACHE, IXLLIST, IXLLSTC, IXLLSTE, IXLLSTM, or IXLLOCK, as appropriate to the type of structure to which you've connected.

  When initially connecting to a structure, you have the option of specifying how long SVC dump holds serialization for a structure when a dump for the connected structure occurs (ACCESSTIME). You also can specify the level of coupling facility that your application desires (CFLEVEL) as well as the minimum CFLEVEL required (MINCFLEVEL).
- **Providing Modification Information**

  Your connection can support rebuilding the structure (ALLOWREBLD) and altering the structure (ALLOWALTER). If the connection is to a cache structure, your connection might also support duplexing the structure (ALLOWDUPREBLD).

– Structure rebuild is a process intended for planned reconfiguration and recovery scenarios, in which connectors to a structure allocate another structure with the same name and rebuild data (if applicable) into the new structure.
– Structure alter is a process that allows a structure’s size, entry-to-element ratio, and, if applicable, percent of structure storage allocated to event monitor controls to be changed. For changes to structure size and entry-to-element ratio, structure alter requires that the structure be allocated in a coupling facility with CFLEVEL=1 or higher. For event monitor control related changes, the list structure must be allocated in a coupling facility with CFLEVEL=3 or higher for a change in size and entry-to-element ratio, or in a coupling facility with CFLEVEL=4 or higher for a change to the percentage of event monitor controls, and the connector(s) must be running on a system with the appropriate level of system upgrades.
– Duplexing rebuild is a process that allows duplexing of a cache structure to be managed by the structure’s connectors.

Starting with OS/390 Release 8, your connection can also support the use of system-managed processes (ALLOWAUTO). When a process is system-managed, the system (as opposed to the connectors) performs some, if not all, of the significant steps in the process. Release 8 provides system-managed rebuild processing. Its primary use is in a planned reconfiguration scenario, for example when moving structures from one coupling facility to another. Even when an application does not support user-managed rebuild (ALLOWREBLD=NO), system-managed rebuild can be used to accomplish the movement of the structures.

Starting with OS/390 Release 10, the installation can specify through its CFRM policy that system-initiated alter processing is requested for a structure. ALLOWAUTOALT(YES) in the CFRM policy indicates that when the structure reaches a specified percent-full threshold, the system is to automatically begin alter processing to relieve the structure storage shortage and also the reclaiming criterion.

IBM recommends that connectors support the structure alter process (ALLOWALTER=YES) because it allows the installation to specify the automatic alter function for the structure. The system then can monitor and tune the structure size and ratios as needed, within the bounds provided by the installation.

IBM also recommends that connectors that support system-managed processes (ALLOWAUTO=YES) should also support the structure alter process (ALLOWALTER=YES). This gives the system the greatest flexibility in allocating new structures during system-managed processes and provides the other benefits that go along with structure alter in general.

The security administrator may have controlled the use of installation structures through the use of RACF or another security product. If so, ensure that you are authorized to issue the IXLCONN macro for the structure.

For information about connection services, see *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established

Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Restrictions

- You cannot issue the IXLCONN macro from the master address space.
- The maximum number of connections to a structure in the coupling facility depends on:
  - The number of CONNECT records in the active CFRM policy.

    The default number of CONNECT records you can specify is 32; the maximum allowed is 255.
  - The limit imposed by the coupling facility control code.

    This is a model-dependent limit. Coupling facilities of CFLEVEL=0 or 1 allow up to 32 connections for all structure types. A CFLEVEL=2 or higher coupling facility allows up to 255 connections for a cache structure and 32 connections for a list or lock structure.
  - The address space limit of 64 serialized structures per address space. (The lock structure and the serialized list structure are "serialized structures".)
  - The application-specified value of the NUMUSERS keyword on a request to connect to a lock structure.

Input Register Information

Before issuing the IXLCONN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLCONN macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if applicable, if GPR 15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLCONN macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLCONN. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

The IXLYCONA macro provides the format of the area that the ANSAREA parameter points to. Include the IXLYCONA macro in your program.

Performance Implications

IXLCONN connect processing involves updating the CFRM active policy to reflect the new connection, allocating the structure in the coupling facility, and/or connecting the user to the structure in the coupling facility. The IXLCONN invoker is suspended while the new connection is notified of all peer connections, as well as all peer connections being notified of the new connection.

Understanding IXLCONN Version Support

The IXLCONN macro supports versions in the range of 0 - 8.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXLCONN macro.
- The following keywords and functions are supported by version 1 and subsequent versions of the IXLCONN macro.

ALLOWALTER	MINENTRY
MINELEMENT	RATIO

- The following keyword is supported by version 2 and subsequent versions of the IXLCONN macro.

EMCSTGPCT

- The following keywords are supported by version 3 and subsequent versions of the IXLCONN macro.

CONNECTIVITY	RNAMELEN

- The following keyword is supported by version 4 and subsequent versions of the IXLCONN macro.

MINEMC

- The following keywords are supported by version 5 and subsequent versions of the IXLCONN macro.

ALLOWDUPREBLD	UDFORDER

- The following keyword is supported by version 6 and subsequent versions of the IXLCONN macro.

NAMECLASSMASK

- The following keywords are supported by version 7 and subsequent versions of the IXLCONN macro.

ALLOWAUTO	SUSPEND

- The following keywords are supported by version 8 and subsequent versions of the IXLCONN macro.

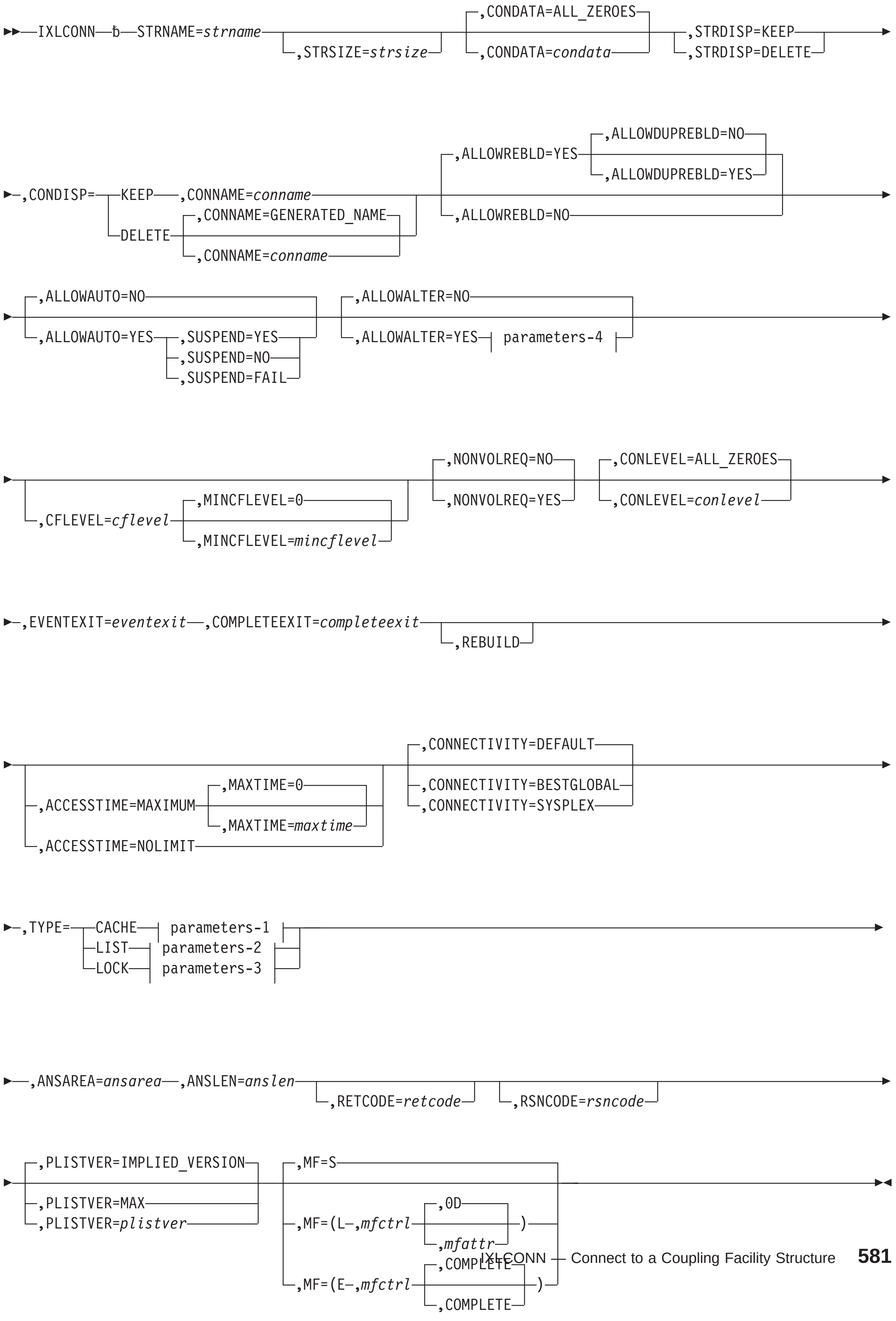
ENTRYIDTYPE	MINCFLEVEL
KEYTYPE	

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLCONN macro is as follows:

main diagram



parameters-1

```
                                                   ┌─,MAXELEMNUM=16─────────┐   ┌─,DIRRATIO=1──────────┐
►►──┬─────────────────────────────────────────┬────┼────────────────────────┼───┼──────────────────────┼──────────────────────────►
    │   ┌─,ELEMCHAR=0──────────────┐          │    └─,MAXELEMNUM=maxelemnum─┘   └─,DIRRATIO=dirratio───┘
    ├───┼──────────────────────────┼──────────┤
    │   └─,ELEMCHAR=elemchar───────┘          │
    └─,ELEMINCRNUM=elemincrnum────────────────┘

   ┌─,ELEMENTRATIO=1──────────────┐  ┌─,ADJUNCT=NO──┐
►──┼──────────────────────────────┼──┼──────────────┼──,VECTORLEN=vectorlen──,NUMCOCLASS=numcoclass──────────►
   └─,ELEMENTRATIO=elementratio───┘  └─,ADJUNCT=YES─┘

                                 ┌─,UDFORDER=NO──┐  ┌─,NAMECLASSMASK=0──────────────┐
►──,NUMSTGCLASS=numstgclass──────┼───────────────┼──┼───────────────────────────────┼──────────────────────────►
                                 └─,UDFORDER=YES─┘  └─,NAMECLASSMASK=nameclassmask──┘

   ┌─,SUPPRESSEVENTS=NO──┐
►──┼─────────────────────┼──────────────────────────────────────────────────────────────────────────────────►◄
   └─,SUPPRESSEVENTS=YES─┘
```

parameters-2

```
                                                   ┌─,MAXELEMNUM=16─────────┐   ┌─,EMCSTGPCT=0───────────┐
►►──┬─────────────────────────────────────────┬────┼────────────────────────┼───┼────────────────────────┼────────────────────────►
    │   ┌─,ELEMCHAR=0──────────────┐          │    └─,MAXELEMNUM=maxelemnum─┘   └─,EMCSTGPCT=emcstgpct───┘
    ├───┼──────────────────────────┼──────────┤
    │   └─,ELEMCHAR=elemchar───────┘          │
    └─,ELEMINCRNUM=elemincrnum────────────────┘

   ┌─,ENTRYRATIO=1──────────────┐  ┌─,ELEMENTRATIO=1──────────────┐  ┌─,ADJUNCT=NO──┐
►──┼────────────────────────────┼──┼──────────────────────────────┼──┼──────────────┼──────────────────────────►
   └─,ENTRYRATIO=entryratio─────┘  └─,ELEMENTRATIO=elementratio───┘  └─,ADJUNCT=YES─┘

   ┌─,LISTCNTLTYPE=ENTRY────┐   ┌─,REFOPTION=NOKEYNAME─────────────────────────┐  ┌─,ENTRYIDTYPE=SYSTEM─┐
►──┼────────────────────────┼───┼──────────────────────────────────────────────┼──┼─────────────────────┼────────►
   └─,LISTCNTLTYPE=ELEMENT──┘   │                   ┌─,KEYTYPE=ENTRY──────┐    │  └─,ENTRYIDTYPE=USER───┘
                                ├─,REFOPTION=KEY────┼─────────────────────┼────┤
                                │                   └─,KEYTYPE=SECONDARY──┘    │
                                └─,REFOPTION=NAME──────────────────────────────┘

   ┌─,VECTORLEN=0─────────────────────────────────────────────┐
►──┼──────────────────────────────────────────────────────────┼──,LISTHEADERS=listheaders─────────────────────►
   │                         ┌─,LISTTRANEXIT=0──────────────┐ │
   └─,VECTORLEN=vectorlen────┼──────────────────────────────┼─┘
                             └─,LISTTRANEXIT=listtranexit───┘

►──┬──────────────────────────────────────────────────────────┬─────────────────────────────────────────────►◄
   └─,LOCKENTRIES=lockentries──,NOTIFYEXIT=notifyexit───────┘
```

parameters-3

```
   ┌─,RECORD=NO──┐  ┌─,RNAMELEN=FIXED64─┐
►►─┼─────────────┼──┼───────────────────┼─,LOCKENTRIES=lockentries─,NUMUSERS=numusers──────────►
   └─,RECORD=YES─┘  └─,RNAMELEN=VAR300──┘

►─,CONTEXIT=contexit─,NOTIFYEXIT=notifyexit───────────────────────────────────────────────────►◄
```

parameters-4

```
   ┌─,RATIO=YES─┐  ┌─,MINENTRY=25────────┐  ┌─,MINELEMENT=25──────────┐  ┌─,MINEMC=25──────┐
►►─┼────────────┼──┼─────────────────────┼──┼─────────────────────────┼──┼─────────────────┼───────►◄
   └─,RATIO=NO──┘  └─,MINENTRY=minentry──┘  └─,MINELEMENT=minelement──┘  └─,MINEMC=minemc──┘
```

Parameter Descriptions

The parameters that are common to all structure types follow in alphabetical order. The additional parameters specific to each structure type are described in “Parameters for TYPE=CACHE” on page 594, “Parameters for TYPE=LIST” on page 597, and “Parameters for TYPE=LOCK” on page 601. Default values are underlined:

Parameters Common to All Structure Types

,ACCESSTIME=MAXIMUM
,ACCESSTIME=NOLIMIT

Use this input parameter to specify the length of time SVC dump holds serialization for the structure when a dump of the structure is requested. This indicates the length of time that a connector can tolerate not having access to the structure. When SVC dump holds serialization on a structure, the system denies new connections to the structure and delays new requests from existing connections to access the structure.

If the user codes ACCESSTIME=MAXIMUM, the maximum time that serialization is held is determined by the first connection that allocates the structure with the value specified for MAXTIME.

If the user codes ACCESSTIME=NOLIMIT, serialization is held for as long as it is required to capture the data from SVC dump or until the structure dump or its serialization would be forced with either the IXLFORCE macro or the SETXCF FORCE command.

ACCESSTIME is set by the first connector to the structure. In the IXLYCONA answer area, the fields CONAACCESSTIME AND CONAACCESSTIMENOLIMIT contain the values pertinent to the allocation of the structure.

Note: SVC dump is supported for cache, list, and serialized list structures only. ACCESSTIME is not valid for lock structures.

,ALLOWALTER=NO
,ALLOWALTER=YES

Use this input parameter to indicate whether this connection allows structure alter to be initiated against the structure. Structure alter can be used to expand or contract the size of a structure, to reapportion the entry-to-element ratio, and to reapportion the event monitor controls (EMC) storage percentage. One or more of these changes can be requested concurrently.

The structure alter procedure is done in place on the previously allocated structure. Structure alter provides a non-disruptive alternative to rebuild for changes to be made to the structure. Changes other than those listed, such as changing the location of a structure, must be accomplished with the rebuild function.

The MINENTRY, MINELEMENT, and MINEMC parameters are used when a structure alter is initiated to contract a structure or reapportion its ratio and/or its EMC storage, where that request would decrease the amount of storage available for entries, elements, or EMCs. These values prevent a structure alter from making a structure unusable to the application, as might occur if more free entries, elements, or EMC storage were removed or reallocated than the application could tolerate. If a structure alter tries to contract or reapportion more free space than specified by the MIN parms, the alter will be stopped. The MIN values are not used during requests to expand the structure, or during reapportion requests which increase the amount of storage available for a particular storage type.

IBM recommends that connectors that support system-managed processes (ALLOWAUTO=YES) should also support the structure alter process (ALLOWALTER=YES). This gives the system the greatest flexibility in allocating new structures during system-managed processes.

NO
Indicates that this connection does not support structure alter for the structure.

YES
Indicates that this connection does support structure alter for the structure. When specifying ALLOWALTER=YES, you must also specify CFLEVEL=1 or higher.

,ALLOWAUTO=NO
,ALLOWAUTO=YES
Use this input parameter to indicate whether this connection supports system-managed processes (for example, rebuild).

When a system-managed rebuild is in progress, any request to connect to the structure will be rejected.

Refer to the IXLREBLD documentation for a description of system-managed rebuild and the effect of the ALLOWAUTO keyword on initiating rebuild.

NO
Indicates that this connection does not support system-managed processes.

An application that does not support structure rebuild (ALLOWREBLD=NO) and specifies or defaults to ALLOWAUTO=NO must provide its own support for planned reconfiguration of the coupling facility. The planned reconfiguration support might include normal shut-down procedures or an operator command interface. The application support must allow the operator to stop an application's use of a structure in a coupling facility.

YES
Indicates that this connection supports system-managed processes.

The application support for system-managed processes includes the following:

- Processing the Structure Temporarily Unavailable and Structure Available events.
- Using extended restart tokens.
- Evaluating the information presented with the Structure State Change event and reacting appropriately to changes in structure characteristics.
- Using both physical structure version numbers to uniquely identify an instance of a structure. (In IXLYCONA, the physical version numbers are identified by CONAPHYSICALSTRUCTUREVERSION and CONAPHYSICALSTRUCTUREVERSION2; in IXLYEEPL, the physical version numbers are identified by EEPLSSCSTRPHYSICALVERSION and EEPLSSCSTRPHYSICALVERSION2).

,ALLOWDUPREBLD=NO
,ALLOWDUPREBLD=YES
Use this input parameter to indicate whether this connection supports user-managed duplexing rebuild. This keyword applies only to cache structures; the keyword is ignored when the structure type is list or lock. This keyword is used in conjunction with the DUPLEX specification for the structure in the active CFRM policy.

If any active or failed-persistent connection to the structure specified ALLOWDUPREBLD=NO, the system will reject a request to start duplexing the structure.

When a duplexing rebuild is in progress, if any new request to connect to the structure specifies ALLOWDUPREBLD=NO, the system will reject the connection request.

Duplexing rebuild allows connectors to a structure to allocate another structure with the same name as the original structure. It allows connectors to duplex data into both structures. Use the duplexing rebuild procedure for improved availability and useability.

If ALLOWDUPREBLD=YES is specified, a protocol for duplexing events must be supported in the event exits of connectors participating in the duplexing rebuild process.

NO
Indicates that this connection does not support duplexing rebuild for the structure.

YES
Indicates that this connection does support duplexing rebuild for the structure.

,ALLOWREBLD=YES
,ALLOWREBLD=NO
Use this input parameter to indicate whether this connection allows structure rebuild to be initiated against this structure. Structure rebuild allows connectors to a structure to allocate another structure with the same name as the original structure, and to rebuild data in the new structure, if applicable. Structure rebuild can be used to change the location and certain attributes of a structure. Use the structure rebuild procedure for planned reconfiguration and for recovery.

If any active connection to the structure specified ALLOWREBLD=NO and specified or defaulted to ALLOWAUTO=NO, the system will reject a request to start rebuilding the structure.

If all active connections specified or defaulted to ALLOWREBLD=YES, the specification of ALLOWAUTO is not considered.

Note that if structure rebuild is allowed, a protocol for rebuild events must be supported in the event exits of connectors participating in the rebuild process.

YES
Indicates that this connection supports structure rebuild for the structure. Refer to the IXLREBLD documentation for a description of user-managed rebuild and the effect of the ALLOWREBLD keyword on initiating rebuild.

NO
Indicates that this connection does not support structure rebuild for the structure. If an application chooses not to support structure rebuild (specifies ALLOWREBLD=NO and specifies or defaults to ALLOWAUTO=NO), then the application must provide its own interfaces for planned reconfiguration of the coupling facility. The planned reconfiguration support might include normal shut-down procedures or an operator command interface. The application support must allow the operator to stop an application's use of a structure in a coupling facility.

,ANSAREA=*ansarea*
Use this output parameter to identify the answer area. When IXLCONN completes, the macro returns information to the answer area. The requestor can use the IXLYCONA macro to map the answer area. ANSAREA must begin on a double-word boundary.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword output field that identifies the answer area.

,ANSLEN=*anslen*
Use this input parameter to specify the length of the answer area. The length should be long enough to accommodate the IXLYCONA mapping of the answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the length of the answer area.

,CFLEVEL=*cflevel*
Use this input parameter to specify that the connector wants the structure to be allocated in a coupling

facility that supports (at least) the indicated level. The connector should request the highest CFLEVEL necessary to process the coupling facility requests the connector anticipates submitting against the structure.

Do not request a level based on whether the connector supports system-managed processes (ALLOWAUTO keyword). It is neither necessary nor desirable to request the CFLEVEL required by a particular system-managed process simply because ALLOWAUTO=YES is specified. (For example, do not specify CFLEVEL=8 for the purpose of supporting system-managed rebuild.) If you specify ALLOWAUTO=YES, the system will automatically attempt to allocate your structure in a coupling facility that supports system-managed processes. Therefore, your CFLEVEL specification should reflect only the level required for the requests that you expect to submit against the structure.

On a successful connect, the field CONACFACILITYCFLEVEL indicates the level of operations that can be performed against the structure. CONAMVSRELEASEMAXCFLEVEL indicates the level of operations supported by the operating system on which the connector is running. These coupling facility levels can differ from the requested level, so the connector must verify that the level returned is suitable for its use. The connector must not issue operations that require a coupling facility level greater than the lower of CONAMVSRELEASEMAXCFLEVEL and CONACFACILITYCFLEVEL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the level of coupling facility required.

,COMPLETEEXIT=*completeexit*
Use this input parameter to identify the complete exit for the requestor. The complete exit receives control in SRB mode, enabled, and unlocked.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the complete exit for the requestor.

,CONDATA=<u>ALL_ZEROES</u>
,CONDATA=*condata*
Use this input parameter to specify eight bytes of connector data. The data is supplied to all of the connector's exits and has no meaning to the system.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 8-character input field that contains the connector data.

,CONDISP=KEEP
,CONDISP=DELETE
Use this input parameter to specify the persistence of the connection after the requestor abnormally terminates or disconnects with REASON=FAILURE.

KEEP
Indicates that the connection to the structure is to be placed in a failed-persistent state. Peer connectors, notified of the Disconnected/Failed Connection event through their event exits, must respond either with the event exit return code or IXLEERSP.

After all peer connectors have responded to the event, XES uses the CONDISP specification to determine whether to make the connector be failed-persistent or not defined. If peer connectors were able to recover for the failing user, they may override the CONDISP=KEEP parameter to indicate that the connector does not need to be made failed-persistent.

Once a connector is placed in a failed-persistent state, the failed connector can try to reconnect to the structure by reissuing IXLCONN with the same CONNAME specified on the original IXLCONN macro. (See CONNAME.)

If the user is unable to reconnect or chooses not to do so, the following options exist to delete the failed-persistent connection:
- The operator or extended MCS console interface can issue the SETXCF FORCE command to delete the connection.
- A connected user or authorized program can issue the IXLFORCE macro to delete the connection.

DELETE
Indicates that the connection to the structure is to be put in the not-defined state after the requestor disconnects or terminates abnormally. Prior to being placed in the not-defined state, peer connectors must have been notified of the Disconnected/Failed Connection event and have responded to confirm the failure.

,CONLEVEL=ALL_ZEROES
,CONLEVEL=*conlevel*
Use this input parameter to specify the connector's processing level that is to be communicated to peer connectors through event exit invocation. The level can indicate a version or release level, a functionality level, or any other connector-defined option.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character input field to contain the connector's processing level.

,CONNAME=GENERATED_NAME
,CONNAME=*conname*
Use this input parameter to specify the unique connection name to identify the user request. CONNAME must be unique for a given structure.

The connection name must be 16 characters long, padded on the right with blanks, if necessary. The name can contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_). Connection names must begin with an uppercase alphabetic character or a national character.

Connection names must be unique for each user connected to a structure. If CONNAME for the request matches the name of another active connection, the system rejects the request.

CONNAME is required for CONDISP=KEEP. To reconnect, a requestor can issue IXLCONN with CONNAME. If the CONNAME matches the name of the failed-persistent connection, then the connector will be reconnected. When a requestor reconnects to the structure, the system returns a new connect token to the connect answer area and sets return code X'4' that the user has reconnected. A reason code of IXLRSNCODESPECIALCONN also is returned to indicate that the connector should examine IXLYCONA to be aware of any changes that might have been made on the reconnect.

CONNAME is optional for CONDISP=DELETE. If a user does not specify CONNAME, the system generates a name. Note however, that CONNAME must be specified when IXLCONN REBUILD is issued, even if CONDISP=DELETE. CONNAME must match the connection name returned in IXLYCONA for the original connection.

If an event that the system reports to the event exit is about a connection to the structure, the system reports the connection's CONNAME to the event exit of all connected users.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the unique connection name to identify the user request.

,CONNECTIVITY=DEFAULT
,CONNECTIVITY=BESTGLOBAL
,CONNECTIVITY=SYSPLEX
Use this input parameter to indicate the required scope of system connectivity to the coupling facility in which the structure is to be allocated.

Use this parameter with the REBUILD option of IXLCONN if you want the system to apply the same allocation rules for a structure that is to be rebuilt as for the initial allocation of the structure. During rebuild processing, the system will select a coupling facility based on the current set of active connectors to the old structure.

CONNECTIVITY=DEFAULT
Specifies that the system is to use the coupling facility selection algorithm to select the coupling facility in which to allocate the structure. The system considers the amount of system connectivity (as specified in the SFM policy, if active) when selecting a coupling facility, but only to decide

between coupling facilities that are equal in all other respects (such as having the requested CFLEVEL and volatility attributes, etc.). If there is no active SFM policy, the system considers each system to be of equal weight.

CONNECTIVITY=BESTGLOBAL
Specifies that the system is to use the expanded coupling facility selection algorithm to select the coupling facility in which to allocate the structure. The system considers the amount of system connectivity before examining all other coupling facility attributes that have been requested. The system attempts to allocate the structure in a coupling facility that has the best connectivity to all systems in the sysplex.

CONNECTIVITY=SYSPLEX
Specifies that the system is to chose a coupling facility for structure allocation that is connected to all systems in the sysplex, whether other requested structure attributes are available or not. If the system cannot identify a coupling facility with connectivity to all systems in the sysplex, the IXLCONN request fails.

,EVENTEXIT=*eventexit*
Use this input parameter to identify the event exit for the requestor. The event exit receives control in SRB mode, enabled, and unlocked.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the event exit for the requestor.

,MAXTIME=0
,MAXTIME=*maxtime*
Use this input parameter to specify the length of time in tenths of seconds that SVC dump holds serialization for the structure when the user specifies ACCESSTIME=MAXIMUM.

Notes:
1. MAXTIME is not valid for lock structures.
2. If the user specifies zero, SVC dump does not obtain serialization for the structure and dump data is not reported for the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword input field that contains the length of time that SVC dump holds serialization for the structure when the user specifies ACCESSTIME=MAXIMUM.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MINCFLEVEL=0
,MINCFLEVEL=*mincflevel*
Use this input parameter to specify that the connector requires that the structure be allocated in a coupling facility that supports at least the indicated minimum CFLEVEL.

The value specified by MINCFLEVEL must be equal to or less than the value specified by CFLEVEL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the minimum CFLEVEL requested.

,MINELEMENT=25
,MINELEMENT=*minelement*
Use this input parameter to specify the minimum level (as a percent) of “in-use” elements that should exist at the end of the structure alter process. For a list structure, this value is a percent of the “currently-in-use” elements. For a cache structure, this value is a percent of the “currently-in-use-and-changed” elements.

When the specified minimum level is attained, IXLALTER processing will end. If the minimum level cannot be attained, IXLALTER processing will provide a percentage of available elements as close to the target level as possible.

The value of MINELEMENT must be in the range of from 0 to 100.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the one-byte field that contains the percent value.

,MINEMC=25
,MINEMC=*minemc*
Use this input parameter to specify the minimum level (as a percent) of “in-use” EMCs that should exist at the end of the alter process. For a keyed list structure, this value is a percent of “currently in-use” EMCs.

When the specified minimum level is attained, IXLALTER processing will end. If the minimum level cannot be attained, IXLALTER processing will provide a percentage of available EMCs as close to the target level as possible.

The value of MINEMC must be in the range of 0 to 100, inclusive.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the one-byte field that contains the percent value.

,MINENTRY=25
,MINENTRY=*minentry*
Use this input parameter to specify the minimum level (as a percent) of “in-use” entries that should

exist at the end of the structure alter process. For a list structure, this value is a percent of the "currently-in-use" entries. For a cache structure, this value is a percent of the "currently-in-use-and-changed" entries.

When the specified minimum level is attained, IXLALTER processing will end. If the minimum level cannot be attained, IXLALTER processing will provide a percentage of available entries as close to the target level as possible.

The value of MINENTRY must be in the range of from 0 to 100.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the one-byte field that contains the percent value.

,NONVOLREQ=NO
,NONVOLREQ=YES

Use this input parameter to indicate whether the connector's use of the structure requires that the coupling facility in which the structure is to be allocated must be both nonvolatile and failure-independent.

The nonvolatility attribute refers to the ability of certain coupling facilities to maintain their stored data should a power outage occur.

The failure-independent attribute refers to the configuration of a coupling facility in an independent failure domain from the MVS systems that access it. An independent failure domain ensures that there is not a single point of failure.

If there is not a coupling facility which is both nonvolatile and failure-independent, XES gives higher priority to allocating the structure in a nonvolatile coupling facility.

NO

Indicates that the system can allocate the structure in a coupling facility regardless of the volatility and failure-independent attributes of the coupling facility.

YES

Indicates that the system must attempt to allocate the structure in a coupling facility that is both nonvolatile and failure-independent. After allocation, the connected user must check the CONASTRUCTUREATTRFLAGS field in the connect answer area (IXLYCONA) to determine the allocated structure's nonvolatility and failure-independent attributes.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLCONN Version Support" on page 580 for a description of the options available with PLISTVER.

,RATIO=YES
,RATIO=NO

Use this input parameter to indicate whether this connection allows changes to the entry-to-element ratio and the event monitor control (EMC) storage percentage when a structure alter is initiated by an IXLALTER request.

YES

Indicates that this connection does allow changes to the entry-to-element ratio and the EMC storage percentage when a structure alter is initiated by an IXLALTER request. When specifying RATIO=YES, you must also specify ALLOWALTER=YES and CFLEVEL=1 or higher.

NO

Indicates that this connection does not allow changes to the entry-to-element ratio and the EMC storage percentage when a structure alter is initiated by an IXLALTER request.

Specifying RATIO=NO does not mean that the entry to element ratio cannot change. It may change as a result of a structure size change, or if the actual structure size

(CONASTRUCTURESIZE) is less than the minimum control space (CONAMINSTRUCTURESIZE) required to allocate the structure with the attributes specified.

,REBUILD

Use this input parameter to specify that the requestor is participating in a rebuilding process for the structure.

The rebuild process allows users to allocate another structure with the same name and to reconstruct data, if applicable, in the newly allocated structure. The rebuild process is intended for planned reconfiguration and recovery scenarios, and requires established protocols to ensure an orderly transition from the old to the new structure.

The requestor that issues IXLCONN with the REBUILD option must be connected to the original structure that is the target of the rebuilding process. If the rebuilding process for the structure has not been initiated (through the SETXCF START,REBUILD command or the IXLREBLD macro), the system rejects the IXLCONN REBUILD request.

If rebuild has been initiated but the new structure has not been allocated, the IXLCONN REBUILD request allocates the new structure. With this parameter, the user must specify the same name (STRNAME) as that of the original structure, and can also modify the following IXLCONN parameters:

- For a cache structure:

ADJUNCT	ELEMENTRATIO	NUMSTGCLASS
CONNECTIVITY	MAXELEMNUM	UDFORDER
DIRRATIO	NAMECLASSMASK	VECTORLEN
ELEMCHAR	NUMCOCLASS	

- For a list structure:

ADJUNCT	EMCSTGPCT	LOCKENTRIES
CONNECTIVITY	ENTRYRATIO	MAXELEMNUM
ELEMCHAR	LISTCNTLTYPE	REFOPTION
ELEMENTRATIO	LISTHEADERS	VECTORLEN

- For a lock structure:

CONNECTIVITY
LOCKENTRIES
NUMUSERS (value must be equal to or greater than the value for the original structure)

The following IXLCONN parameters can be specified, but the system ignores any changes made to the values in effect from the original invocation of IXLCONN for the structure:

ALLOWALTER	CONDISP	SUSPEND
ALLOWAUTO	CONLEVEL	EVENTEXIT
ALLOWDUPREBLD	MINELEMENT	COMPLETEEXIT
ALLOWREBUILD	MINEMC	LISTTRANEXIT (if required)
CFLEVEL	MINENTRY	NOTIFYEXIT (if required)
CONDATA	RATIO	CONTEXIT (if required)
	STRDISP	

Once the new structure has been allocated, REBUILD requests from subsequent users connect the users to the new structure. As with any request to connect to a structure, the connectors must check the connect answer area to determine the actual attributes of the structure.

REBUILD requests must be issued from the same address space and from the same system as that from which the original IXLCONN macro for the structure was issued. REBUILD requests can be issued from a task other than the task that issued IXLCONN for the original structure.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,STRDISP=KEEP
,STRDISP=DELETE
Use this input parameter to specify the persistence of the structure when there are no longer any connectors, either active or failed.

KEEP
Indicates that the structure remains allocated after all connectors have disconnected from the structure.

DELETE
Indicates that the structure is deleted from the coupling facility after all connectors (including failed-persistent connections) have disconnected from the structure.

STRNAME=*strname*
Use this input parameter to specify the name of the structure that you want to connect to in the coupling facility. The structure must be defined in the active CFRM policy at the time of the connect. You can use the IXCQUERY macro to determine which structures have been defined in the active CFRM policy.

The structure name must be 16 characters long, padded on the right with blanks, if necessary. The name can contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_). Structure names must begin with an uppercase alphabetic character. IBM structure names begin with SYS, an IBM component prefix, and the letter A through I.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the structure name.

,STRSIZE=0
,STRSIZE=*strsize*
Use this input parameter to specify the size of the structure in units of 4K blocks. The system allocates the structure following these guidelines:

- If the size specified is **smaller** than the minimum structure size that is specified or defaulted to in the active CFRM policy (MINSIZE), the system uses the MINSIZE value as the size of the structure.
- If the size specified is **smaller** than the structure size defined in the active CFRM policy (SIZE), the system uses the *strsize* value. The *strsize* is bounded by MINSIZE and SIZE in the active CFRM policy.
- If the size specified is **larger** than the structure size defined in the active CFRM policy, the system uses the size defined in the active policy (SIZE from policy).
- If you use the default value of zero for *strsize*, the system uses the size defined in the active CFRM policy (INITSIZE, if specified, or SIZE).

Note that in all cases, if there are limited coupling facility resources available, the system might allocate the structure using less space than was requested. MINSIZE specified in the active CFRM policy will serve as a minimum bound for the structure on all structure allocation requests (including alter, connect, and rebuild connect) except for new structure allocation during System-managed rebuild. New structure allocation during System-managed rebuild may cause the structure to be allocated with a size smaller than MINSIZE in the active CFRM policy. The actual structure size allocated is returned in field CONASTRUCTURESIZE in the IXLYCONA answer area. The requested structure size is returned in field CONAALLOCREQUESTEDSTRSIZE in IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of 4K blocks to make up the structure.

,SUSPEND=YES
,SUSPEND=NO
,SUSPEND=FAIL

Use this input parameter to indicate whether the connection wants the system to:

- Suspend work units that issue coupling facility requests against a structure while it is undergoing a system-managed process (for example, rebuild), regardless of the MODE specified on the coupling facility request.
- Process such requests as indicated by the MODE specification of the request.
- Fail such requests.

The SUSPEND keyword does not affect coupling facility requests that specify MODE=SYNCFAIL.

YES

Indicates that the system will, when possible, override a request's MODE specification and suspend the requestor during a system-managed process. Specifying SUSPEND=YES permits the system to limit the number of incoming requests by quiescing the work units that would otherwise be submitting them. This minimizes the system resources required to quiesce activity against the structure undergoing the system-managed process.

Note that when the requestor is resumed, the system will notify the requestor of request completion in the manner specified by the original MODE. If the MODE requested that the system attempt to complete the request synchronously (for example, SYNCEXIT), the requestor will receive a return code indicating synchronous completion. If the MODE requested asynchronous processing, (for example, ASYNCEXIT), the requestor will receive a return code indicating asynchronous completion and will be notified of the results of the request through the specified mechanism (for example, the Complete Exit).

NO

Indicates that the connector cannot tolerate suspension of units of work submitting coupling facility requests except as specified on the MODE keyword of list (IXLLIST, IXLLSTC, IXLLSTE, IXLLSTM) and cache (IXLCACHE) requests and will therefore use suspend/resume processing only on units of work that specify MODE=SYNCSUSPEND. The system will quiesce all other activity by deferring requests internally until the system-managed process completes.

FAIL

Indicates that the connector cannot tolerate the potentially long-term suspension or delay of units of work submitting coupling facility requests while the structure is quiesced for system-managed processing. Requests which cannot be immediately processed due to the structure being quiesced for system-managed processing will be failed. Such requests will neither be deferred internally for asynchronous processing, nor will the requesting unit of work be suspended. If the structure is quiesced to fall out of a system-managed duplexing rebuild, which is only a short-term delay, then the request will be processed as if SUSPEND=NO had been specified. SUSPEND=FAIL is not applicable to lock or serialized list structures. Connect attempts to these types of structures will fail if SUSPEND=FAIL is specified.

,TYPE=CACHE

,TYPE=LIST
,TYPE=LOCK

Use this input parameter to identify the type of structure in the coupling facility to which you want to connect. The parameter descriptions applicable to a cache structure follow. For parameter descriptions applicable to a list structure, see “Parameters for TYPE=LIST” on page 597; for parameter descriptions applicable to a lock structure, see “Parameters for TYPE=LOCK” on page 601.

Parameters for TYPE=CACHE

,ADJUNCT=NO
,ADJUNCT=YES

Use this input parameter to specify whether adjunct data areas are associated with each cache entry in the coupling facility cache structure. Each adjunct data area is 64 bytes.

NO
Indicates that adjunct data areas should not be allocated for the cache structure.

YES
Indicates that adjunct data areas should be allocated for each entry in the cache structure.

,DIRRATIO=1
,DIRRATIO=*dirratio*

Use this input parameter to specify a value to express the directory portion of the directory-to-element ratio of the coupling facility cache structure. The value of DIRRATIO must be greater than zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte input field that contains a value to express the directory portion of the directory-to-element ratio of the coupling facility cache structure.

,ELEMCHAR=0
,ELEMCHAR=*elemchar*

Use this input parameter to specify the value of an element characteristic used to determine the element size. The element size is calculated with the formula 256*(2**ELEMCHAR), where ELEMCHAR is used as the power of 2. For example, if ELEMCHAR=0, then the size of each element is 256 bytes.

The valid values for ELEMCHAR range from zero to a maximum determined by the coupling facility limitation on element size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the value of the element characteristic, used to determine the cache structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the cache structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element characteristic. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,ELEMENTRATIO=1
,ELEMENTRATIO=*elementratio*

Use this input parameter to specify a value to express the element portion of the directory-to-element ratio of the coupling facility cache structure. If ELEMENTRATIO is zero, the structure is allocated without data elements.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte input field that contains a value to express the element portion of the directory-to-element ratio of the coupling facility cache structure.

,ELEMINCRNUM=*elemincrnum*

Use this input parameter to specify the element increment number. The element size is calculated with the formula 256*ELEMINCRNUM. For example, if ELEMINCRNUM=1, then the size of each element is 256 bytes.

The valid values for ELEMINCRNUM range from 1 to a maximum determined by the coupling facility limitation on element size. The value of ELEMINCRNUM must be a power of 2.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the value of the element increment number, used to determine the cache structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the cache structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element increment number. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,MAXELEMNUM=16
,MAXELEMNUM=*maxelemnum*

Use this input parameter to specify a value to determine the maximum number of data elements for each data entry in the coupling facility cache structure.

- With a coupling facility of CFLEVEL=0 or higher, you can specify from 1 to 16 for MAXELEMNUM.
- With a coupling facility of CFLEVEL=1 or higher, you can specify from 1 to 255 for MAXELEMNUM.

The maximum data entry size in bytes equals MAXELEMNUM multiplied by the element size obtained from the value specified for ELEMCHAR or ELEMINCRNUM.

For example, if ELEMCHAR=0 and MAXELEMNUM=1, the maximum data entry size in bytes is 256. If ELEMCHAR=4 and MAXELEMNUM=16, the maximum data entry size in bytes is 4096(16), or 65,536.

To ensure that the system can assign all possible data elements allocated in a structure to a directory entry, MAXELEMNUM must be greater than or equal to ELEMENTRATIO divided by DIRRATIO. MAXELEMNUM is ignored if ELEMENTRATIO is zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the maximum number of data elements for each data entry in the coupling facility cache structure.

,NAMECLASSMASK=0
,NAMECLASSMASK=*nameclassmask*

Use this input parameter to specify the name class mask pattern definition to be applied to entry names for the purpose of assigning entries to name classes in the structure. Name classes may be used to improve the efficiency of processing for commands such as IXLCACHE REQUEST=DELETE_NAME.

The position of each bit in the name class mask corresponds to the same relative character position in an entry name. If a bit in the name class mask is a one, then the corresponding position in the entry name is applied in the masking operation to determine the name class to which the entry will be assigned. If a bit is a zero, then the corresponding position in the entry name is not applied to assigning the entry to a name class.

Specifying or defaulting to a name class mask of 0 (X'0000'), or specifying a name class mask of all binary ones (X'FFFF'), will result in name classes not being used for the structure. A CFLEVEL of 7 or greater must be specified when any other name class mask is specified, implying that name classes are to be used.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-bit input field that contains the name class mask pattern definition.

,NUMCOCLASS=*numcoclass*
Use this input parameter to specify the number of cast-out classes for named data entries used with the coupling facility cache structure. You must specify at least 1 cast-out class.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of cast-out classes.

,NUMSTGCLASS=*numstgclass*
Use this input parameter to specify the number of storage classes for named data entries used with the coupling facility cache structure. You must specify at least 1 storage class.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of storage classes.

,SUPPRESSEVENTS=NO
,SUPPRESSEVENTS=YES
Use this input parameter to indicate whether the origination of user connection events (New Connection, Existing Connection, Rebuild New Connection, Rebuild Existing Connection), and user disconnection/failure events (Disconnected or Failed Connection), should be suppressed for this connection. Suppression of these events may provide a significant performance benefit at connect or disconnect time to connectors who do not need the information presented in these events.

Note that suppression of these events is on a per-connection basis, where it is the ORIGINATION of the event that is suppressed, not the RECEIPT of the event. Therefore, the following cases are possible:

- A connection that is suppressing the origination of events may nevertheless receive such events for another connection to the structure that is NOT suppressing the origination of events. In the case of response-required events, this connection is still required to provide responses to any such event that is presented to it, regardless of whether it is suppressing events itself.
- When a connection is suppressing the origination of events, other connections will not receive the suppressed events for that connection, even if the other connections themselves are NOT suppressing events.

Also note that, depending on whether the system on which the connection is running is at OS/390 Release 9 or higher or has OW38840 installed, a request to suppress events may or may not be honored for the current connection.

NO
Indicates that connection and disconnection related events will be originated normally for this connection.

YES
Indicates that the origination of connection and disconnection related events may be suppressed for this connection. (Dependent on the level of the system on which the connection is running.)

,UDFORDER=NO
,UDFORDER=YES
Use this input parameter to indicate whether a user data field (UDF) order queue should be maintained for each cast-out class for the structure. A coupling facility with CFLEVEL=5 or higher is required when UDFORDER=YES.

Depending on whether UDF-order queues are maintained, a READ_COSTATS request to retrieve cast-out class statistics will include the following information:

- If a UDF-order queue is maintained, the system will return the user data field of the first entry on the UDF-order queue.
- If a UDF-order queue is not maintained, the system will return the user data field of the first entry on the cast-out class queue.

,VECTORLEN=*vectorlen*

Use this input parameter to specify the number of cache buffers in the requestor's local storage which require concurrent registration. The requestor uses the vector length to map each local cache buffer to a named data entry in the coupling facility cache structure. The value of VECTORLEN must be a multiple of 32 or the system will round the value up to a multiple of 32.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of cache buffers in the requestor's local storage.

Parameters for TYPE=LIST

,ADJUNCT=NO
,ADJUNCT=YES

Use this input parameter to specify whether adjunct data areas are associated with each list entry in the coupling facility list structure. Each adjunct data area is 64 bytes.

NO

Indicates that there are no adjunct data areas specified for the list structure.

YES

Indicates that each list entry in the list structure has an associated adjunct data area of 64 bytes.

,ELEMCHAR=0
,ELEMCHAR=*elemchar*

Use this input parameter to specify the value of an element characteristic used to determine the element size. The element size is calculated with the formula 256*(2**ELEMCHAR), where ELEMCHAR is used as the power of 2. For example, if ELEMCHAR=0, then the size of each element is 256 bytes.

The valid values for ELEMCHAR range from zero to a maximum determined by the coupling facility model limitation on element size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte field that contains the value of the element characteristic, used to determine the list structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the list structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element characteristic. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,ELEMENTRATIO=1
,ELEMENTRATIO=*elementratio.*

Use this input parameter to specify a value to express the data element portion of the entry-to-element ratio of the coupling facility list structure. If ELEMENTRATIO is zero, the list structure is allocated without data elements. If ELEMENTRATIO is zero, then ADJUNCT must equal YES.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte field that contains a value to express the element portion of the entry-to-element ratio of the coupling facility list structure.

,ELEMINCRNUM=*elemincrnum*

Use this input parameter to specify the element increment number. The element size is calculated with the formula 256*ELEMINCRNUM. For example, if ELEMINCRNUM=1, then the size of each element is 256 bytes.

The valid values for ELEMINCRNUM range from 1 to a maximum determined by the coupling facility model limitation on element size. The value of ELEMINCRNUM must be a power of 2.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte field that contains the value of the element increment number, used to determine the list structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the list structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element increment number. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,EMCSTGPCT=0
,EMCSTGPCT=*emcstgpct*

Use this input parameter to specify the percentage of available list structure storage that is to be set aside for event monitor controls (EMCs). The structure must be a keyed list structure (REFOPTION=KEY) and its allocation must be requested in a coupling facility with CFLEVEL=3 or higher.

Specify the percentage value in hundredths of a percent. For example, to request a value of 20%, code EMCSTGPCT=2000.

The value of EMCSTGPCT must be in the range of from 0 to 10000.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword field that contains a value to express the percentage of available structure storage that is to be set aside for event monitor controls (EMCs).

,ENTRYIDTYPE=SYSTEM
,ENTRYIDTYPE=USER

Use this input parameter to specify whether the system or the user will assign the list entry IDs for list entries created in this structure. A list entry can be located by its assigned entry ID. A connect request will be rejected if the connector's requested ENTRYIDTYPE attribute does not match the attribute in effect for the currently allocated instance of the structure. A rebuild connect request will be rejected if the connector's requested ENTRYIDTYPE attribute does not match the attribute in effect for the original structure.

ENTRYIDTYPE=SYSTEM

The system will assign the list entry IDs for list entries created in this structure. An entry ID will be generated by the system for each list entry created, and it will be unique among the list entries previously and currently allocated in the list structure. A system generated entry ID is never reused and is unique for the life of the structure.

ENTRYIDTYPE=USER

The user will assign the entry ID for each list entry created in this structure. An entry ID must be provided by the user for each list entry created, and it must be unique among the list entries currently allocated in the list structure. A user-provided entry ID may be reused over time provided it is unique while assigned to a list entry.

A CFLEVEL of 8 or higher must also be specified when ENTRYIDTYPE=USER is specified. Note that an ENTRYIDTYPE=USER request will be rejected if a coupling facility of CFLEVEL=8 or higher is not available in which to allocate the structure.

,ENTRYRATIO=1
,ENTRYRATIO=*entryratio*

Use this input parameter to specify a value to express the list entry portion of the entry-to-element ratio of the coupling facility list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte field that contains a value to express the list entry portion of the entry-to-element ratio of the coupling facility list structure.

,KEYTYPE=ENTRY
,KEYTYPE=SECONDARY

Use this input parameter to specify whether only entry keys, or both entry keys and secondary keys may be used when processing or locating list entries, or when comparing the entry keys of list entries.

KEYTYPE=ENTRY

Only entry keys may be used when processing or locating list entries, or when comparing the entry keys of list entries.

Each list entry is assigned a 16-byte entry key when it is created. The user may provide the entry key, specify that the list key value be assigned, or allow the entry key value to be assigned a default value. The entry key may be reassigned when the entry is moved. Event monitoring is provided for keyed sublists.

See the IXLLSTC, IXLLSTE, and IXLLSTM macros for additional information.

KEYTYPE=SECONDARY

Both entry keys and secondary keys may be used when processing or locating list entries, or when comparing the entry keys of list entries.

Entry keys are stored in the list entry, whereas secondary keys are stored in the first 32 bytes of the 64-byte list entry adjunct area, thus restricting the available user space in the adjunct area to the second 32 bytes. Note also that the list structure must be allocated to use list entries with adjunct data. Thus, in addition to KEYTYPE=SECONDARY, ADJUNCT=YES must also be specified on the IXLCONN invocation when allocating the structure and when establishing a connection with the structure.

Each list entry is assigned a 16-byte entry key and a 32-byte secondary key when it is created. The user may provide the secondary key or allow the secondary key value to be assigned a default value. While the entry key can be reassigned by any of the move requests, the secondary key can be reassigned only by an IXLLSTM REQUEST=MOVE_ENTRYLIST invocation. Event monitoring is provided for entry keyed and secondary keyed sublists.

See the IXLLSTC, IXLLSTC, and IXLLSTM macros for additional information.

ADJUNCT=YES and a CFLEVEL of 9 or greater must also be specified when KEYTYPE=SECONDARY is specified.

KEYTYPE=SECONDARY is meaningful only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

,LISTCNTLTYPE=ENTRY
,LISTCNTLTYPE=ELEMENT

Use this input parameter to specify whether the system maintains and tracks list limits based on number of entries or number of elements.

ENTRY

Specifies that the list limits are specified and tracked as limits on the number of entries which may reside on the list.

ELEMENT

Specifies that the list limits are specified and tracked as limits on the total number of data elements which may be associated with entries on the list.

,LISTHEADERS=*listheaders*
Use this input parameter to specify the number of lists (list headers) for the coupling facility list structure. The number must be greater than zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of list headers to be allocated.

,LISTTRANEXIT=*listtranexit*
Use this input parameter to identify the list transition exit for the requestor.

The list transition exit is used to inform users when one or more lists or their event queue they are monitoring changes from the empty state to the non-empty state.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the list transition exit for the requestor.

,LOCKENTRIES=0
,LOCKENTRIES=*lockentries*
Use this input parameter to specify the number of lock entries for the coupling facility list structure. If the value for the number of lock entries is not a power of 2, it is rounded upward to the nearest power of 2. If the value is not specified, or if the requestor specifies zero, then the system does not support serialization for the allocated list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the number of lock entries for the coupling facility list structure.

,MAXELEMNUM=16
,MAXELEMNUM=*maxelemnum*
Use this input parameter to specify a value to determine the maximum number of elements for each list entry in the coupling facility list structure.

- With a coupling facility of CFLEVEL=0 or higher, you can specify from 1 to 16 for MAXELEMNUM.
- With a coupling facility of CFLEVEL=1 or higher, you can specify from 1 to 255 for MAXELEMNUM.

The maximum list entry size in bytes equals MAXELEMNUM multiplied by the element size obtained from the value specified for ELEMCHAR or ELEMINCRNUM.

For example, if ELEMCHAR=0 and MAXELEMNUM=1, the maximum list entry size in bytes is 256. If ELEMCHAR=4 and MAXELEMNUM=16, the maximum list entry size in bytes is 4096(16), or 65,536.

To ensure that the system can assign all possible data elements allocated in a structure to a list entry, MAXELEMNUM must be greater than or equal to ELEMENTRATIO divided by ENTRYRATIO. MAXELEMNUM is ignored if ELEMENTRATIO is zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte field that contains a value to determine the maximum number of elements for each list entry in the coupling facility list structure.

,NOTIFYEXIT=*notifyexit*
Use this input parameter to identify the notify exit for the requestor.

The notify exit is used to inform a list services (IXLLIST, IXLLSTC, IXLLSTE, IXLLSTM) user that contention exists for a lock held by the user.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the notify exit for the requestor.

,REFOPTION=NOKEYNAME
,REFOPTION=KEY
,REFOPTION=NAME
Use this input parameter to specify how to reference list entries in the coupling facility list structure. The requestor can specify one of the following:

NOKEYNAME
Indicates that the list entry can be located by the list entry identifier (LEID). The system assigns an LEID for each list entry that is in use in the coupling facility list structure. Neither keys nor names are used to reference list entries.

KEY
Indicates that the list entry can be located by a key value. When creating the entry, the requestor can assign a key value to one or more list entries on the macro.

NAME
Indicates that the list entry can be located by a unique name. When creating the entry, the requestor can assign a unique name to each list entry on the list services (IXLLIST, IXLLSTC, IXLLSTE, IXLLSTM) macro.

,VECTORLEN=0
,VECTORLEN=*vectorlen.*
Use this input parameter to specify the number of vector entries in the vector that is to be used to monitor state transitions for list headers or the event queue. A list header or an event queue undergoes a state transition when it changes from an empty to nonempty state, or from a nonempty to an empty state.

The value of VECTORLEN must be a multiple of 32 or the system will round the value up to a multiple of 32.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the number of vector entries in the vector that is to be used to monitor state transitions for list headers or the event queue.

Parameters for TYPE=LOCK

,CONTEXIT=*contexit*
Use this input parameter to identify the contention exit for the requestor.

The contention exit is used to resolve resource contention based on user-defined protocols.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the contention exit for the requestor.

,LOCKENTRIES=*lockentries*
Use this input parameter to specify the number of lock entries to be allocated for the coupling facility lock structure. If the value for LOCKENTRIES is not a power of 2, it is rounded upward to the nearest power of 2. If a lock structure contains record data (RECORD=YES), you must specify a nonzero value for LOCKENTRIES. If a lock structure does not contain record data (RECORD=NO) and the value for LOCKENTRIES is zero, the system attempts to allocate the structure with as many lock entries as possible within the given structure size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the number of lock entries to be allocated for the lock structure. If the lock structure supports record data, a value of zero for LOCKENTRIES is not valid. If the lock structure does not support record data, a value of zero indicates that the system should obtain the largest possible number of lock entries given the structure's allocated size.

,NOTIFYEXIT=*notifyexit*
Use this input parameter to specify the name of the notify exit for the requestor.

The notify exit is used to notify resource owners that contention for the resource exists.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the name of the notify exit for the requestor.

,NUMUSERS=*numusers*
Use this input parameter to specify the maximum number of users that may connect to and use the lock structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte field that contains the maximum number of users of the coupling facility lock structure.

,RECORD=NO
,RECORD=YES

Use this input parameter to specify whether the lock user wants to do recording of data about the lock:

NO
Indicates the system does not record data.

YES
Indicates the system records data.

,RNAMELEN=FIXED64
,RNAMELEN=VAR300

Use this input parameter to indicate the length attribute of the resource names (either fixed length or variable length). All connectors to the structure must specify the same attribute when they connect to the structure. The system rejects a connect request if the connector's requested RNAMELEN value does not match the attribute in effect for a currently-allocated instance of the structure, as determined by the IXLCONN RNAMELEN specification of the first connector to the structure. The system also rejects a rebuild connect request if RNAMELEN is specified and its value is not consistent with the value specified for the original structure.

RNAMELEN=FIXED64
Specifies that the resource names for the structure will have a fixed length of 64 bytes.

RNAMELEN=VAR300
Specifies that the resource names for the structure will have a variable length between 1 and 300 bytes.

Return and Reason Codes

When the IXLCONN macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. xxxx indicates internal information.

Table 40. Return and Reason Codes for the IXLCONN Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** Processing completed successfully. The system returns data to ANSAREA. To map ANSAREA, use IXLYCONA. **Action:** None. It is the responsibility of the connector to verify that the structure attributes, as recorded in the connect answer area, are acceptable.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0407	**Equate Symbol:** IXLRSNCODESPECIALCONN **Meaning:** The connection was completed. The CONA provides additional status information about the structure and the connector. The field CONAFLAGS indicates any special conditions that affect the connection to the structure. If any of the CONAFLAGS bits are ON, the connector receives this reason code. CONAFLAGS contains flags which indicate one or more of the following: connector has been reconnected, rebuild in progress, rebuild stop in progress, alter in progress, or a user sync point event is set. **Action:** If required, interrogate the CONAFLAGS field to determine the status of the structure. It is the responsibility of the connector to verify that the structure attributes, as recorded in the connect answer area, are acceptable.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The IXLCONN parameter list is not accessible. **Action:** Verify that • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are calling IXLCONN in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLCONN in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLCONN.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** Program error. The IXLCONN parameter list ALET is not valid. The request fails. **Action:** Ensure that the ALET is zero or that the ALET represents a valid entry on the DU-AL.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** Program error. Invalid version number in the IXLCONN parameter list. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS your program is running on.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** Program error. The requestor is in SRB mode. **Action:** Do not issue the IXLCONN macro when running in SRB mode.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. The requestor is not in an enabled state. **Action:** Issue the IXLCONN macro while running enabled for I/O and external interrupts.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0808	**Equate Symbol:** IXLRSNCODEMASTERAS **Meaning:** Program error. The connection is not permitted from the Master address space. **Action:** Do not issue the IXLCONN macro from the Master address space.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Program error. The primary address space does not equal the home address space. **Action:** Make sure that the primary address space and the home address space are the same at the time of the IXLCONN invocation.
8	xxxx080D	**Equate Symbol:** IXLRSNCODEAREATOOSMALL **Meaning:** Program error. ANSAREA is too small, as specified by ANSLEN. **Action:** Ensure that the ANSLEN parameter properly reflects the size of the area specified by ANSAREA. Also, make sure that the length of ANSAREA is large enough to contain the data returned.
8	xxxx080E	**Equate Symbol:** IXLRSNCODEBADAREA **Meaning:** Program error. Unable to access ANSAREA. **Action:** Make sure that ANSAREA is properly addressed prior to the IXLCONN invocation.
8	xxxx080F	**Equate Symbol:** IXLRSNCODEBADAREAALET **Meaning:** Program error. The ANSAREA ALET is not valid. **Action:** Verify that the ANSAREA ALET is valid.
8	xxxx081B	**Equate Symbol:** IXLRSNCODENOLENTRIES **Meaning:** Program error. The number of lock entries specified for a lock structure with record data was zero. **Action:** Ensure that if a lock structure is to support record data, the value of LOCKENTRIES is greater than zero.
8	xxxx081C	**Equate Symbol:** IXLRSNCODENOLISTHDRS **Meaning:** Program error. The number of list headers specified for LISTHEADERS must be greater than zero. **Action:** Ensure that the IXLCONN LISTHEADERS parameter is a value greater than zero.
8	xxxx081D	**Equate Symbol:** IXLRSNCODEZEROLUSERS **Meaning:** Program error. The number of users specified for TYPE=LOCK NUMUSERS must be greater than zero. **Action:** Ensure that the IXLCONN NUMUSERS parameter value is greater than zero.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx081F	**Equate Symbol:** IXLRSNCODECONNAME **Meaning:** Program error. CONNAME matches the CONNAME of another active connection to the same structure. CONNAME must be unique for each connection to a structure. **Action:** Make sure that the CONNAME specified is not already connected to the structure.
8	xxxx0820	**Equate Symbol:** IXLRSNCODESTRTYPE **Meaning:** Program error. The structure type specified does not match the type of the currently-allocated structure or the RNAMELEN attribute specified does not match that of the currently-allocated structure. When you connect to an allocated structure or issue a rebuild connect to a structure, you cannot change some of the structure attributes, specifically TYPE and RNAMELEN. The request fails. **Action:** Specify a value for TYPE or for RNAMELEN that matches the attribute specified when the structure was allocated.
8	xxxx0821	**Equate Symbol:** IXLRSNCODESTRSERIAL **Meaning:** Program error. The serialization attribute for a list structure specified by the LOCKENTRIES keyword on connect does not match the currently-allocated structure. When you connect to an allocated structure, you cannot change the structure attributes. **Action:** Specify a value for LOCKENTRIES that matches that specified for original structure.
8	xxxx0823	**Equate Symbol:** IXLRSNCODECONNAMEERR **Meaning:** Program error. CONNAME is not valid. **Action:** Verify that the CONNAME meets the requirements of the keyword.
8	xxxx084C	**Equate Symbol:** IXLRSNCODENOSAFAUTH **Meaning:** Program error. Connector does not have the proper SAF authorization. In IXLYCONA, CONADIAG1 contains the SAF return code and CONADIAG2 contains the SAF reason code. **Action:** Determine why the installation has not allowed the proper SAF authorization for access to the structure.
8	xxxx0856	**Equate Symbol:** IXLRSNCODEREBUILDNUMUSER **Meaning:** Program error. The value for NUMUSERS on IXLCONN REBUILD is less than the value for NUMUSERS specified on the IXLCONN request for the original structure. The request fails. **Action:** Specify a value for NUMUSERS on the IXLCONN REBUILD that matches that specified for the original structure.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx085B	**Equate Symbol:** IXLRSNCODERECORDLISTATTR **Meaning:** Program error. The record data attribute for the structure being rebuilt is not the same as the record data for the original structure. **Action:** Specify a value for RECORD on the IXLCONN REBUILD that matches that specified for the original structure.
8	xxxx085C	**Equate Symbol:** IXLRSNCODEINVALIDLISTATTR **Meaning:** Program error. A list structure must be allocated with one of the following: lock entries, data elements, or adjunct data. None were specified. **Action:** Specify the request with at least one of the following: lock entries, data elements, or adjunct data.
8	xxxx085D	**Equate Symbol:** IXLRSNCODEINVALIDSTGCLASS **Meaning:** Program error. NUMSTGCLASS for a cache structure cannot equal zero. **Action:** Specify a value for NUMSTGCLASS that is greater than zero.
8	xxxx085E	**Equate Symbol:** IXLRSNCODEINVALIDCOCLASS **Meaning:** Program error. NUMCOCLASS for a cache structure cannot equal zero. **Action:** Specify a value for NUMCOCLASS that is greater than zero.
8	xxxx085F	**Equate Symbol:** IXLRSNCODEINVALIDVECTORLEN **Meaning:** Program error. VECTORLEN for a cache structure cannot equal zero. The request fails. **Action:** Specify a value for VECTORLEN that is non-zero for a cache structure.
8	xxxx0860	**Equate Symbol:** IXLRSNCODEDIRRATIO **Meaning:** Program error. DIRRATIO for a cache structure cannot equal zero. Directory entries are required for a cache structure. **Action:** Specify a value for DIRRATIO that is greater than zero.
8	xxxx0861	**Equate Symbol:** IXLRSNCODEENTRYRATIO **Meaning:** Program error. When ELEMENTRATIO is greater than zero, ENTRYRATIO must be greater than zero. List entry controls are required for a list structure. **Action:** Specify a value for ENTRYRATIO that is greater than zero.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0862	**Equate Symbol:** IXLRSNCODEMAXELEMNUM **Meaning:** Program error. • LIST Structure: If ELEMENTRATIO is not zero, then MAXELEMNUM must be greater than or equal to ELEMENTRATIO divided by ENTRYRATIO when allocating a list structure. • CACHE Structure: If ELEMENTRATIO is not zero, the MAXELEMNUM for must be greater than or equal to ELEMENTRATIO divided by DIRRATIO when allocating a cache structure. **Action:** Modify the value of MAXELEMNUM to meet the required conditions.
8	xxxx0863	**Equate Symbol:** IXLRSNCODETASKTERM **Meaning:** Program error. The IXLCONN request is rejected because the requesting task is going through task termination. IXLCONN cannot be issued from a resource manager. **Action:** Examine your protocol to ensure that the IXLCONN macro is not issued from a resource manager.
8	xxxx086B	**Equate Symbol:** IXLRSNCODEELEMINCRNUM **Meaning:** Program error. The value specified for ELEMINCRNUM is not valid. It must be non-zero and be a power of 2. **Action:** Modify the value of ELEMINCRNUM to meet the required conditions.
8	xxxx086F	**Equate Symbol:** IXLRSNCODEINVALIDCFLEVEL **Meaning:** Program error. The request to alter a structure is rejected because ALLOWALTER=YES was specified and a CFLEVEL of zero was either specified or taken as a default. A CFLEVEL of 1 or higher is required when ALLOWALTER=YES is specified. **Action:** Ensure that CFLEVEL=1 is specified.
8	xxxx0870	**Equate Symbol:** IXLRSNCODEREBUILDVECTORLEN **Meaning:** Program error. The VECTORLEN value specified for the rebuild structure is not consistent with the VECTORLEN attribute of the original structure. **Action:** If you specified VECTORLEN on the IXLCONN request for the original structure, then you must also specify it on the IXLCONN request for the rebuild structure. Or, if you did not specify VECTORLEN on the IXLCONN request for the original structure, then you must not specify it on the IXLCONN request for the rebuild structure.
8	xxxx0871	**Equate Symbol:** IXLRSNCODEMAXELEMNUMELEMCHAR **Meaning:** Program error. The values specified in MAXELEMNUM and ELEMCHAR would result in the structure having greater than 64K entries. **Action:** Modify the value(s) of MAXELEMNUM and/or ELEMCHAR to meet the 64K limit.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0872	**Equate Symbol:** IXLRSNCODEMINENTRY **Meaning:** Program error. The value specified for MINENTRY is not valid. **Action:** Modify the value of MINENTRY to be within the range of from 0 to 100.
8	xxxx0873	**Equate Symbol:** IXLRSNCODEMINELEMENT **Meaning:** Program error. The value specified in MINELEMENT is not valid. **Action:** Modify the value of MINELEMENT to be within the range of from 0 to 100.

8	xxxx0881	**Equate Symbol:** IXLRSNCODEBADCFLEVEL
		
		Meaning: Program error. The request is rejected because the specified parameter is not appropriate for the value specified for CFLEVEL. The specified reason is indicated in the following table. The reason code index is returned in the CONADIAG2 field of the connect answer area mapped by IXLYCONA.
		
		`Rsn Code    Specified          Minimum`
		`Index       Parameter          CFLEVEL`
		
		`   1       EMCSTGPCT>0           3`
		`   2       UDFORDER=YES             5`
		`   3       NAMECLASSMASK            7`
		`   4       ENTRYIDTYPE=USER         8`
		`   5       KEYTYPE=SECONDARY        9`
		
		Action: Correct the parameter value in error.

8	xxxx0882	**Equate Symbol:** IXLRSNCODEBADREFOPTION **Meaning:** Program error. The request is rejected because the specified parameter is not appropriate for the value specified for REFOPTION. **For PARAMETER** — **REFOPTION must be** **EMCSTGPCT>0** — KEY **Action:** Correct the parameter value in error.
8	xxxx0883	**Equate Symbol:** IXLRSNCODEBADEMCSTGPCT **Meaning:** Program error. The value specified for EMCSTGPCT is not valid. The value must be between 0 and 10000. **Action:** Correct the value of EMCSTGPCT.
8	xxxx0884	**Equate Symbol:** IXLRSNCODEBADMINEMC **Meaning:** Program error. The value specified for MINEMC is not valid. The value must be between 0 and 100. **Action:** Correct the value of MINEMC.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx088D	**Equate Symbol:** IXLRSNCODEBADENTRYIDTYPE **Meaning:** Program error. The requested ENTRYIDTYPE is not consistent with the ENTRYIDTYPE attribute of the allocated structure. Either ENTRYIDTYPE=USER was requested but the structure was previously allocated with ENTRYIDTYPE=SYSTEM, or ENTRYIDTYPE=SYSTEM was requested and the structure was previously allocated with ENTRYIDTYPE=USER. **Action:** Correct the value of ENTRYIDTYPE.

8	xxxx088E	**Equate Symbol:** IXLRSNCODEINCONSISTENTPARM
		
		Meaning: Program error. A keyword specification was made that also requires one or more other keywords to be specified. The specific reason is indicated in the following table. The reason code index is returned in the CONADIAG2 field of the connect answer area mapped by IXLYCONA.
		`Rsn Code          Specified              Also`
		`Index             Parameter             Required`
		
		`   1           KEYTYPE=SECONDARY        ADJUNCT=YES`
		
		Action: Correct the parameter value in error.

8	xxxx089E	**Equate Symbol:** IXLRSNCODEBADMINCFLEVEL **Meaning:** Program error. Request rejected. The specified MINCFLEVEL value is greater than the specified CFLEVEL value. **Action:** Correct the value for MINCFLEVEL or CFLEVEL as appropriate.
8	xxxx08A9	**Equate Symbol:** IXLRSNCODEBADSUSPENDOPTION **Meaning:** Program error. Request rejected. SUSPEND=FAIL is not valid for lock or serialized list structures. **Action:** Correct the value either of the SUSPEND parameter.
C	xxxx0C02	**Equate Symbol:** IXLRSNCODENOMORECONNS **Meaning:** Environmental error. The structure has the maximum number of permitted connections. **Action:** Retry the request at a later time. The ENF signal 35 will be signalled when coupling facility resources become available. The maximum number of permitted connections may be limited by the value of PLEXCFG.
C	xxxx0C04	**Equate Symbol:** IXLRSNCODEJOINFAILED **Meaning:** Environmental error. IXCJOIN has failed. (XES invokes IXCJOIN on behalf of a user of a serialized structure (TYPE=LOCK or TYPE=LIST with LOCKENTRIES). Additional diagnosis information exists in the connect answer area, IXLYCONA. The field CONADIAG1 contains the IXCJOIN return code and CONADIAG2 contains the IXCJOIN reason code. **Action:** Use the IXCJOIN return and reason codes to understand why IXCJOIN failed and attempt to resolve the problem.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C05	**Equate Symbol:** IXLRSNCODENOTINPOLICY **Meaning:** Environmental error. The active CFRM policy for the installation does not specify the structure. The request fails. **Action:** Specify a structure that is currently defined in the CFRM active policy or switch to a new CFRM policy that contains the structure for which this IXLCONN was invoked.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. The system does not have connectivity to the coupling facility containing the specified structure. This may occur due to operator commands such as VARY PATH OFFLINE or CONFIG CHP OFFLINE or hardware errors such as coupling facility or path failures. **Action:** Determine why connectivity was not established and re-attempt the request. In IXLYCONA, the field CONACONNECTORCONNECTIVITY is set if at least one active connector has connectivity to the structure.
C	xxxx0C08	**Equate Symbol:** IXLRSNCODENOFAC **Meaning:** Environmental error. Allocation of the structure failed because there was no suitable coupling facility to allocate the structure based on the preference list in the active CFRM policy. **Action:** In IXLYCONA, the field CONAFACILITYARRAY contains information about each coupling facility in which XES attempted to allocate the structure. Examine each reason code associated with each coupling facility to understand the reason why the allocation failed for that coupling facility.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C09	**Equate Symbol:** IXLRSNCODECONNPREVENTED **Meaning:** Environmental error. New connections to the requested structure are being prevented at this time for one of the following reasons: • All active connectors have confirmed either the Rebuild Quiesce event or the Duplex Complete event. New connections will not be permitted until the rebuild stop or rebuild cleanup is completed. • The structure is allocated in a coupling facility that is failed. New connections will not be permitted until the structure is rebuilt, or all connections disconnect causing the structure to be deallocated. • The coupling facility containing the structure is not available for use because policy reconciliation is in progress. New connections will not be permitted until policy reconciliation is complete. • New structure allocations for this structure name are not permitted because there is a pending policy change for this structure. New connections will not be permitted until the structure cleanup is complete. • Structure cleanup is in progress. New connections will not be permitted until the structure cleanup is complete. • A system-managed process (for example, system-managed rebuild) is in progress. When the structure is in the Duplex Established phase of a system-managed duplexing rebuild, CONASTRUCTURESMDUPESTAB flag will be set. If the user specified or defaulted to ALLOWAUTO=NO on the connect, the connect will be prevented as long as the structure remains duplexed. For each of the reasons listed, an ENF Event Code 35 will be signalled when the condition preventing new connections has completed. **Action:** Determine why connection to the structure was prevented and retry the request at a later time.
C	xxxx0C0C	**Equate Symbol:** IXLRSNCODENOCONNDUPLEXNEWSTR **Meaning:** Environmental error. This system does not have connectivity to the coupling facility containing the specified duplexed new structure. This may occur because of an operator command such as VARY PATH,OFFLINE or CONFIG CHP,OFFLINE or hardware errors such as coupling facility or channel path failures. MVS stops the structure duplexing rebuild for the structure. **Action:** Determine why connectivity was not established and re-attempt the request. In IXLYCONA, the field CONACONNECTORCONNECTIVITY is set if at least one active connector has connectivity to the structure.
C	xxxx0C11	**Equate Symbol:** IXLRSNCODEDEFINE **Meaning:** Environmental error. The local vector requested on connect could not be defined on a TYPE=CACHE or TYPE=LIST request. **Action:** Retry the request at a later time.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C12	**Equate Symbol:** IXLRSNCODEDSNOTCREATED **Meaning:** Environmental error. The system could not create a data space. **Action:** Retry the request at a later time.
C	xxxx0C15	**Equate Symbol:** IXLRSNCODEMAXCONNECTAS **Meaning:** Environmental error. The requestor has exceeded the maximum number of serialized connections (limit of 64) for the address space. **Action:** Retry the request at a later time. Examine your protocol to determine why the maximum has been exceeded.
C	xxxx0C16	**Equate Symbol:** IXLRSNCODEPASNEXCEEDED **Meaning:** Environmental error. An error occurred adding to the PASN access list. **Action:** Retry the request at a later time.
C	xxxx0C19	**Equate Symbol:** IXLRSNCODETIMERNOTSET **Meaning:** Environmental error. XES DIE could not be established for this system. Specifically, the XES DIE could not be established for this system. **Action:** Retry the request at a later time.
C	xxxx0C23	**Equate Symbol:** IXLRSNCODECONNOTINPOL **Meaning:** Environmental error. The request failed because there is a failed-persistent connection with the same connection name that has not been reconciled into the CFRM policy. **Action:** Examine your active CFRM policy. The CFRM couple data set requires additional CONNECT records. This might require that the CFRM couple data set be reformatted.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. Request failed because structure failure has occurred. The request fails. **Action:** Retry the request at a later time. The system will issue ENF signal 35 when coupling facility resources become available.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C27	**Equate Symbol:** IXLRSNCODERSPNOTREC **Meaning:** Environmental error. A connector to a structure has either failed or disconnected. All surviving connectors to the structure have not yet responded to the EEPLDISCFAILCONN event. XES cannot perform cleanup processing for a terminating connector until all surviving connectors have responded. The EEPLDISCFAILCONN event allows users to respond with either a return code in IXLYEEPL or with an IXLYEEPL return code and a later IXLEERSP response. Users are not required to use only IXLEERSP when responding. **Action:** Surviving connectors can specify how XES is to perform cleanup processing by responding to the EEPLDISCFAILCONN event in one of three ways: • Set return code X'0' in IXLYEEPL (normal processing) • Set return code X'1' in IXLYEEPL (used for failed-persistent connections) • Set return code X'8' in IXLYEEPL to delay your response and then respond with IXLEERSP. Examine your protocol to determine why responses have not been issued. The CONADISCFAILEDCONFSTRING field of the IXLYCONA contains a string indicating the connections that must provide a response for the disconnect/failure of the previous instance of the connection. Retry the request at a later time. The ENF signal 35 will be signalled when all responses for the disconnect/failure have been received.
C	xxxx0C29	**Equate Symbol:** IXLRSNCODEXESNOTACTIVE **Meaning:** Environmental error. The CFRM function is not active or not available. **Action:** Bring the CFRM couple data set in use in the sysplex by issuing the SETXCF COUPLE,TYPE=CFRM,PCOUPLE= command.
C	xxxx0C30	**Equate Symbol:** IXLRSNCODEDUMPINPROGRESS **Meaning:** Environmental error. The connection could not be completed because SVC Dump holds serialization on the structure. **Action:** Retry the request at a later time. The ENF signal 35 will be signalled when coupling facility resources become available.
C	xxxx0C3C	**Equate Symbol:** IXLRSNCODEREBUILDCONNECT **Meaning:** Environmental error. The rebuild connection failed because the original connector failed (task was terminated) while processing this IXLCONN REBUILD request. This reason code is only applicable when the IXLCONN REBUILD request is issued from a different task than the task that owns the original connection. **Action:** None. The original connect that failed cannot reconnect.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C3D	**Equate Symbol:** IXLRSNCODENOTREBUILDING **Meaning:** Environmental error. The connection failed because the requestor issued IXLCONN REBUILD for a structure that is not rebuilding. **Action:** Verify that the structure name has been specified correctly.
C	xxxx0C44	**Equate Symbol:** IXLRSNCODEREBUILDCONNEXISTS **Meaning:** Environmental error. The connection failed because REBUILD connect already exists for this CONNAME. **Action:** Verify that the connection name has been specified correctly.
C	xxxx0C45	**Equate Symbol:** IXLRSNCODEREBUILDBADCONN **Meaning:** Environmental error. Either the issuer of IXLCONN with the REBUILD option is not a connector in the address space or system from which this request was issued or the connector is not active. **Action:** To issue an IXLCONN REBUILD request, the user must be connected to the original structure and the request must be issued from the same address space and system as the original request.
C	xxxx0C47	**Equate Symbol:** IXLRSNCODEREBUILDCONNPHASE **Meaning:** Environmental error. The connection failed because the requestor issued IXLCONN REBUILD during the wrong phase of the rebuild process. The cause is one of the following: • The IXLCONN REBUILD was issued before all connections have provided a response to the Rebuild Quiesce event. • The IXLCONN REBUILD was issued after all connections have issued IXLREBLD REQUEST=COMPLETE. **Action:** Examine your protocol to determine why the request was issued during the wrong phase of the rebuild process.
C	xxxx0C4E	**Equate Symbol:** IXLRSNCODEREBUILDCONNECTSTOP **Meaning:** Environmental error. The IXLCONN REBUILD request was not successful because a rebuild stop occurred. **Action:** Determine why the rebuild stop occurred.
C	xxxx0C50	**Equate Symbol:** IXLRSNCODEREBUILDCONNECTNOPREF **Meaning:** Environmental error. The IXLCONN REBUILD request was not successful because there is no policy information for this structure in the active CFRM policy. This structure was reconciled into the policy from the coupling facility when the sysplex was IPLed with a down-level CFRM couple data set. **Action:** Switch to a new CFRM active policy that includes the definition for this structure.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C52	**Equate Symbol:** IXLRSNCODEALLOWREBLD **Meaning:** Environmental error. The connect request was rejected because a connector specified either ALLOWREBLD=NO and structure rebuild is in progress or ALLOWDUPREBLD=NO and structure duplexing is in progress. **Action:** Examine your protocol to determine why a user specified that rebuild or duplexing was not to be allowed.
C	xxxx0C53	**Equate Symbol:** IXLRSNCODECFLEVEL **Meaning:** Environmental error. The connect request was not successful because the requestor specified a CFLEVEL value that is greater than the maximum CFLEVEL supported by the MVS release on which the IXLCONN was requested. **Action:** The maximum CFLEVEL supported by the MVS release is returned in the CONAMVSRELEASEMAXCFLEVEL field of the IXLYCONA. Interrogate this field to determine the supported level.
C	xxxx0C64	**Equate Symbol:** IXLRSNCODESTRALTERNOTALLOW **Meaning:** Environmental error. The connect request was rejected because alter is in progress and the connection specified ALLOWALTER=NO on IXLCONN. **Action:** Wait until the structure alter is complete and then retry the connect request. The system issues an ENF event code 35 specifying the name of the structure when the structure alter completes.
C	xxxx0C65	**Equate Symbol:** IXLRSNCODESTRALTERRESTRICT **Meaning:** The connect request was rejected because alter is in progress and the RATIO, MINENTRY, and/or MINELEMENT specification is more restrictive than the current composite for existing connections. **Action:** Wait until the structure alter is complete and then retry the connect request. The system issues an ENF event code 35 specifying the name of the structure when the structure alter completes.
C	xxxx0C9A	**Equate Symbol:** IX.RSNCODEINSUFFCFLEVELUSER **Meaning:** The connect request was rejected because the user specified a MINCFLEVEL value that is greater than the level of the coupling facility in which the target structure is allocated. The level of the coupling facility in which the target structure is allocated is returned in the CONACFACILITYCFLEVEL field of IXLYCONA. **Action:** Determine the level of the coupling facility in which the target structure is allocated. If the IXLCONN request failed because the structure is currently allocated in a coupling facility that is below the program's minimum requirements for CFLEVEL as specified by MINCFLEVEL, have the structure moved to a coupling facility that is at or above that MINCFLEVEL. Attempt the IXLCONN request with the same MINCFLEVEL.

Table 40. Return and Reason Codes for the IXLCONN Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Equate Symbol:** IXLRETCODECOMPERROR **Meaning:** Failure in XES processing. Additional diagnostic information might exist in the following fields in IXLYCONA: CONADIAG0, CONADIAG1, CONADIAG2, CONADIAG6, CONADIAG7, CONADIAG8, and CONADIAG9. **Action:** Provide the IXLYCONA information listed above to the IBM support center.

IXLCSP — XES Structure Computation Service

Description

The IXLCSP service assists in coupling facility capacity planning and exploitation by providing a means to evaluate the following coupling facility structure values:

- Required structure size, given structure attributes and object counts
- Resulting structure object counts, given structure attributes and size.

Calculations do not result in the actual allocation of stuctures, nor do they affect the current contents of the target coupling facility.

Calculations are appropriate to the CFLEVEL of the target coupling facility, but do not take into account current coupling facility conditions such as storage constraints. These conditions might prevent the actual allocation of a structure in the target coupling facility, or might cause the structure to be allocated with significantly different counts than returned by IXLCSP.

You can use the IXLMG service to obtain additional information about the target coupling facility, such as CFLEVEL, model-dependent limits, and storage.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN, any SASN.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established.
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

The caller's parameter list must be addressable from the unit of work issuing the request.

The IXLYCSPA mapping macro provides the format of the area that the ANSAREA points to. Include that macro in your program.

Input Register Information

Before issuing the IXLCSP macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLCSP macro, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code, if applicable, if GPR 15 return code is non-zero
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system

15	Return code

When control returns to the caller of the IXLCSP macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Performance Implications

None.

Understanding IXLCSP Version Support

The IXLCSP macro supports versions 0 and 1.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXLCSP macro.
- The following keyword is supported by version 1 and subsequent versions of the IXLCSP macro.

KEYTYPE

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLCSP macro is as follows:

main diagram

```
►►──IXLCSP──b──CFNAME=cfname──,TYPE=──┬─CACHE──┤ parameters-1 ├─┬──,ANSAREA=ansarea──,ANSLEN=anslen──────►
                                      ├─LIST───┤ parameters-2 ├─┤
                                      └─LOCK───┤ parameters-3 ├─┘

                                                  ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬──┼───────────────────────────┼──────────────────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                                  └─,PLISTVER=plistver────────┘

   ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────────────────┼─────────────────────────────────────────────────────────►◄
   │                 ┌─,0D──────┐          │
   ├─,MF=(L,mfctrl───┼──────────┼─)────────┤
   │                 └─,mfattr──┘          │
   │                 ┌─,COMPLETE─┐         │
   ├─,MF=(M,mfctrl───┼───────────┼─)───────┤
   │                 └─,NOCHECK──┘         │
   │                 ┌─,COMPLETE─┐         │
   └─,MF=(E,mfctrl───┼───────────┼─)───────┘
                     └─,NOCHECK──┘
```

parameters-1

```
                                     ┌─,MAXELEMNUM=16───────────┐
►►──┬──────────────────────────────┬─┴──────────────────────────┴──┬─,ADJUNCT=NO──┬──────────────►
    │ ┌─,ELEMCHAR=1──────────┐     │  └─,MAXELEMNUM=maxelemnum─┘   └─,ADJUNCT=YES─┘
    ├─┴──────────────────────┴─────┤
    │ └─,ELEMCHAR=elemchar───┘     │
    └─,ELEMINCRNUM=elemincrnum─────┘

►─,NUMCOCLASS=numcoclass──,NUMSTGCLASS=numstgclass──┬─,UDFORDER=NO──┬──┬─,NAMECLASS=NO──┬──────►
                                                    └─,UDFORDER=YES─┘  └─,NAMECLASS=YES─┘

►─,REQUEST=──┬─COMPUTECOUNTS──┤ parameters-4 ├─┬────────────────────────────────────────────►◄
             └─COMPUTESIZE──┤ parameters-5 ├───┘
```

parameters-2

```
                                     ┌─,MAXELEMNUM=16───────────┐
►►──┬──────────────────────────────┬─┴──────────────────────────┴──┬─,ADJUNCT=NO──┬──────────────►
    │ ┌─,ELEMCHAR=1──────────┐     │  └─,MAXELEMNUM=maxelemnum─┘   └─,ADJUNCT=YES─┘
    ├─┴──────────────────────┴─────┤
    │ └─,ELEMCHAR=elemchar───┘     │
    └─,ELEMINCRNUM=elemincrnum─────┘

►──┬─,LISTCNTLTYPE=ENTRY───┬──┬─,REFOPTION=NOKEYNAME─────────────────────────┬─,LISTHEADERS=listheaders────►
   └─,LISTCNTLTYPE=ELEMENT─┘  │                    ┌─,KEYTYPE=ENTRY─────┐    │
                              ├─,REFOPTION=KEY─────┴────────────────────┴────┤
                              │                    └─,KEYTYPE=SECONDARY─┘    │
                              └─,REFOPTION=NAME──────────────────────────────┘

   ┌─,LOCKENTRIES=0─────────────┐
►──┴────────────────────────────┴──┬─,PLEIDS=YES─┬──,REQUEST=──┬─COMPUTECOUNTS──┤ parameters-6 ├─┬──────►◄
   └─,LOCKENTRIES=lockentries─┘    └─,PLEIDS=NO──┘             └─COMPUTESIZE──┤ parameters-7 ├───┘
```

parameters-3

```
►►──┬─,RECORD=NO──┬──,LOCKENTRIES=lockentries──,NUMUSERS=numusers──────────────────────────────►
    └─,RECORD=YES─┘

►─,REQUEST=──┬─COMPUTECOUNTS──┤ parameters-8 ├─┬────────────────────────────────────────────►◄
             └─COMPUTESIZE──┤ parameters-9 ├───┘
```

parameters-4

```
    ┌─,DIRRATIO=1──────────┐    ┌─,ELEMENTRATIO=1──────────────┐
►►──┴──────────────────────┴────┴──────────────────────────────┴──┤ parameters-8 ├──────────────►◄
    └─,DIRRATIO=dirratio───┘    └─,ELEMENTRATIO=elementratio───┘
```

parameters-5

```
                                                                 ┌─,MAXSIZE=COMPUTEDSTRSIZE─┐
►►──,DIRENTRYCOUNT=direntrycount──,ELEMENTCOUNT=elementcount─────┴──────────────────────────┴──────►◄
                                                                 └─,MAXSIZE=maxsize─────────┘
```

parameters-6

```
   ┌─,EMCSTGPCT=0──────────────┐ ┌─,ENTRYRATIO=1──────────────┐ ┌─,ELEMENTRATIO=1────────────────┐
►►─┼───────────────────────────┼─┼────────────────────────────┼─┼────────────────────────────────┼────►
   └─,EMCSTGPCT=emcstgpct──────┘ └─,ENTRYRATIO=entryratio─────┘ └─,ELEMENTRATIO=elementratio────┘

►─┤ parameters-8 ├──────────────────────────────────────────────────────────────────────────────►◄
```

parameters-7

```
►►─,EMCCOUNT=emccount─,ENTRYCOUNT=entrycount─,ELEMENTCOUNT=elementcount─────────────────────────►

  ┌─,MAXSIZE=COMPUTEDSTRSIZE─┐
►─┼──────────────────────────┼──────────────────────────────────────────────────────────────────►◄
  └─,MAXSIZE=maxsize─────────┘
```

parameters-8

```
►►─,STRSIZE=strsize─,MAXSIZE=maxsize────────────────────────────────────────────────────────────►◄
```

parameters-9

```
                                    ┌─,MAXSIZE=COMPUTEDSTRSIZE─┐
►►─,RDATENTRYCOUNT=rdatentrycount───┼──────────────────────────┼────────────────────────────────►◄
                                    └─,MAXSIZE=maxsize─────────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ADJUNCT=NO
,ADJUNCT=YES

Use this input parameter to indicate whether adjunct data is associated with each structure entry. Adjunct data is 64 bytes per entry.

NO
The structure entries do not have associated adjunct data.

YES
The structure entries have associated adjunct data.

,ANSAREA=*ansarea*

Use this output area to contain information returned about the request. The storage area is mapped by the IXLYCSPA macro.

Not all fields in the answer area are applicable to all request types.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an output area to contain information about the request.

,ANSLEN=*anslen*

Use this input parameter to specify the length of the answer area. The length should be long enough to accommodate the IXLYCSPA mapping of the answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the length of the answer area.

CFNAME=*cfname*

Use this input parameter to specify the name of the coupling facility for which computations are to be performed. The specified coupling facility must be:

- Defined in the active CFRM policy
- Connected to the system from which the request is issued
- CFLEVEL=8 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character input field that contains the name of the coupling facility.

,DIRENTRYCOUNT=*direntrycount*
Use this input parameter to specify the desired maximum number of directory entries in the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input area to contain the maximum number of directory entries.

,DIRRATIO=<u>1</u>
,DIRRATIO=*dirratio*
Use this input parameter to specify a value to express the directory portion of the directory-to-element ratio of the coupling facility structure. The value of DIRRATIO must be greater than zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte input field that contains a value to express the directory portion of the directory-to-element ratio of the coupling facility structure.

,ELEMCHAR=<u>0</u>
,ELEMCHAR=*elemchar*
Use this input parameter to specify the value of an element characteristic used to determine the element size. The element size is calculated with the formula 256*(2**ELEMCHAR), where ELEMCHAR is used as the power of 2. For example, if ELEMCHAR=0, then the size of each element is 256 bytes.

The valid values for ELEMCHAR range from zero to a maximum determined by the coupling facility limitation on element size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the value of the element characteristic, used to determine the structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element characteristic. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,ELEMENTCOUNT=*elementcount*
Use this input input parameter to specify the desired maximum number of data elements in the structure.

For a directory-only cache structure, or a list structure without data elements, specify ELEMENTCOUNT=0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input area to contain the maximum number of data elements in the structure.

,ELEMENTRATIO=<u>1</u>
,ELEMENTRATIO=*elementratio*
Use this input parameter to specify a value to express the element portion of the directory-to-element ratio for a cache structure or the entry-to-element ratio for a list structure. If ELEMENTRATIO is zero, the calculation reflects a structure allocated without data elements.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte input field that contains a value to express the element portion of the directory-to-element ratio of the coupling facility structure.

,ELEMINCRNUM=*elemincrnum*

Use this input parameter to specify the element increment number. The element size is calculated with the formula 256*ELEMINCRNUM. For example, if ELEMINCRNUM=1, then the size of each element is 256 bytes.

The valid values for ELEMINCRNUM range from 1 to a maximum determined by the coupling facility limitation on element size. The value of ELEMINCRNUM must be a power of 2.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the value of the element increment number, used to determine the cache structure element size.

Note: The element size, in combination with the maximum number of elements (MAXELEMNUM parameter), determines the size of the data entry — the data written to and read from the structure. With a coupling facility of CFLEVEL=0, a data entry can be up to 16 times the data element size, as indicated by the element increment number. With a coupling facility of CFLEVEL=1 or higher, a data entry can be up to 255 times the data element size.

Use either ELEMCHAR or ELEMINCRNUM to define the element size. You can specify either an element characteristic or an element increment number for element size determination.

,EMCCOUNT=*emccount*

Use this input parameter to specify the desired maximum number of event monitor controls in the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input area to contain the maximum number of event monitor controls in the structure.

,EMCSTGPCT=0
,EMCSTGPCT=*emcstgpct*

Use this input parameter to specify the percentage of available list structure storage that is to be set aside for event monitor controls (EMCs). The amount of storage requested is that over and above the storage amount included in the marginal structure size.

Specify the percentage value in hundredths of a percent. For example, to request a value of 20%, code EMCSTGPCT=2000.

The value of EMCSTGPCT must be in the range of 0 to 10000. A non-zero value indicates tha some connector intends to perform sublist monitoring. Note that for sublist monitoring, the structure must be a keyed list structure (REFOPTION=KEY).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the halfword field that contains a value to express the percentage of available structure storage, over and above the storage amount included in the marginal structure size, that is to be set aside for event monitor controls (EMCs).

,ENTRYCOUNT=*entrycount*

Use this input parameter to specify the desired maximum number of list entries in the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input area to contain the maximum number of directory or list entries in the structure.

,ENTRYRATIO=1
,ENTRYRATIO=*entryratio*

Use this input parameter to specify a value to express the list entry portion of the entry-to-element ratio of the coupling facility list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 2-byte field that contains a value to express the list entry portion of the entry-to-element ratio of the coupling facility list structure.

,KEYTYPE=ENTRY
,KEYTYPE=SECONDARY

Use this input parameter to indicate whether keyed list entries will use enty keys and secondary keys or only entry keys.

ENTRY
Specifies that only list entry keys will be used.

SECONDARY
Specifies that list entry keys and secondary list entry keys will be used.

Secondary keys are stored in the first 32 bytes of the 64-byte list entry adjunct area. Therefore, when KEYTYPE=SECONDARY is specified, adjunct data must also be requested by specifying ADJUNCT=YES. KEYTYPE=SECONDARY requires a coupling facility of CFLEVEL=9 or higher.

,LISTCNTLTYPE=ENTRY
,LISTCNTLTYPE=ELEMENT

Use this input parameter to specify whether the system maintains and tracks list limits based on number of entries or number of elements.

ENTRY
Specifies that the list limits are specified and tracked as limits on the number of entries that may reside on the list.

ELEMENT
Specifies that the list limits are specified and tracked as limits on the total number of data elements that may be associated with entries on the list.

,LISTHEADERS=*listheaders*

Use this input parameter to specify the number of lists (list headers) for the coupling facility list structure. The number must be greater than zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of list headers to be allocated.

,LOCKENTRIES=0
,LOCKENTRIES=*lockentries*

Use this input parameter to specify the number of lock entries associated with the structure. If the value for the number of lock entries is not a power of 2, it is rounded upward to the nearest power of 2.

- When specified for a list structure, a LOCKENTRIES value of 0 indicates an unserialized list structure.
- When specified for a lock structure, a LOCKENTRIES value of 0 indicates that IXLCSP is to determine the maximum number of locks that can be accomodated by a structure of the specified size. Specifying LOCKENTRIES=0 is valid only when RECORD=NO is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the number of lock entries for the coupling facility structure.

,MAXELEMNUM=16
,MAXELEMNUM=*maxelemnum*

Use this input parameter to specify a value to determine the maximum number of data elements for each data entry in the coupling facility structure. MAXELEMNUM must be in the decimal range of 1 to 255.

The maximum data entry size in bytes equals MAXELEMNUM multiplied by the element size obtained from the value specified for ELEMCHAR or ELEMINCRNUM.

For example, if ELEMCHAR=0 and MAXELEMNUM=1, the maximum data entry size in bytes is 256. If ELEMCHAR=4 and MAXELEMNUM=16, the maximum data entry size in bytes is 4096(16), or 65,536.

To ensure that the system can assign all possible data elements allocated in a structure to a directory or list entry, MAXELEMNUM must be greater than or equal to:

- ELEMENTRATIO divided by DIRRATIO (for cache structures)
- ELEMENTRATIO divided by ENTRYRATIO (for list structures)

when computing structure counts. MAXELEMNUM is ignored if ELEMENTRATIO is zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte input field that contains the maximum number of data elements for each data entry in the coupling facility structure.

,MAXSIZE=COMPUTEDSTRSIZE
,MAXSIZE=*maxsize*

Use this input parameter to specify the maximum size of the structure in units of 4K blocks. The MAXSIZE parameter corresponds to the SIZE parameter that would be used in defining the structure in the CFRM policy.

If MAXSIZE is specified, and its value is greater than the size computed from the other input parameters, then it is also used as an input to the computation. If MAXSIZE is not specified, the computation uses the computed structure size as the maximum size.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input area to contain the maximum size of the structure in units of 4K blocks.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(M,*mfctrl***)**
,MF=(M,*mfctrl***,COMPLETE)**
,MF=(M,*mfctrl***,NOCHECK)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
,MF=(E,*mfctrl***,NOCHECK)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=M to specify the modify form of the macro. Use the modify form to generate code to put the parameters into the parameter list.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
,NOCHECK
Use this input parameter to specify the degree of macro parameter syntax checking the system is to do.

COMPLETE
Use this parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

NOCHECK
Use this parameter to specify that the system is not to check for required parameters nor to supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,NAMECLASS=NO
,NAMECLASS=YES
Use this input parameter to indicate whether the structure supports name classes. Name classes are supported when the IXLCONN request that results in allocation of the structure specifies a value of NAMECLASSMASK other than X'0000' or X'FFFF'.

NO
The structure does not support name classes.

YES
The structure supports name classes.

,NUMCOCLASS=*numcoclass*
Use this input parameter to specify the desired number of cast-out classes to be used with the coupling facility cache structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of cast-out classes.

,NUMSTGCLASS=*numstgclass*
Use this input parameter to specify the desired number of storage classes used with the coupling facility cache structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the number of storage classes.

,NUMUSERS=*numusers*
Use this input parameter to specify the maximum number of users that may connect to the lock structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 1-byte field that contains the maximum number of connectors to the coupling facility lock structure.

,PLEIDS=YES
,PLEIDS=NO
Use this input parameter to specify whether the list structure is to be allocated with programmable list entry identifiers (PLEIDs). For list structures that are to be used for XCF signalling, specify PLEIDS=NO. For all other list structures, specify PLEIDS=YES.

YES
The calculation will reflect a list structure allocated with PLEIDs.

NO
The calculation will reflect a list structure that is not allocated with PLEIDs.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLCSP Version Support" on page 618 for a description of the options available with PLISTVER.

,RDATENTRYCOUNT=*rdatentrycount*
Use this input parameter to specify the desired number of record data entries in a lock structure.

RDATENTRYCOUNT is ignored when RECORD=NO is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the maximum number of record data entries.

,RECORD=NO
,RECORD=YES
Use this input parameter to specify whether the lock structure has associated record data.

NO
The lock structure does not have record data.

YES
The lock structure does have record data.

,REFOPTION=NOKEYNAME
,REFOPTION=KEY
,REFOPTION=NAME
Use this input parameter to specify how to reference list entries in the coupling facility list structure. The list entry can always be referenced by list entry identifier (LEID) or by unkeyed position. Choose one of the following options to specify how the list entry is to be referenced.

NOKEYNAME
The list entries have neither keys nor names.

KEY
The list entries have keys.

NAME
The list entries have names.

,REQUEST=COMPUTECOUNTS
,REQUEST=COMPUTESIZE
Use this input parameter to specify the function requested.

COMPUTECOUNTS
Compute the number of structure objects that can be contained within a structure of the specified size and attributes.

COMPUTESIZE
Compute the size of a structure having the specified attributes and containing the specified number of objects.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,STRSIZE=*strsize*

Use this input parameter to specify the structure size in units of 4K blocks. The STRSIZE parameter corresponds to the INITSIZE parameter that would be used in defining the structure in the CFRM policy.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field to contain the structure size in units of 4K blocks.

,TYPE=CACHE
,TYPE=LIST
,TYPE=LOCK

Use this input parameter to specify the type of structure for which computations are to be made.

,UDFORDER=NO
,UDFORDER=YES

Use this input parameter to indicate whether a user data field (UDF) order queue should be maintained for each cast-out class for the structure.

NO

The calculation is to reflect a cache structure that does not maintain UDF order queues.

YES

The calculation is to reflect a cache structure that maintains UDF order queues.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the IXLCSP macro returns control to your program:
- GPR 15 (and *retcode*, you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 41. Return and Reason Codes for the IXLCSP Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** IXLCSP request successful. **Action:** None.

Table 41. Return and Reason Codes for the IXLCSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list is either not addressable or not accessible. **Action:** Verify that the parameter list address is valid.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** Program error. The parameter list ALET does not represent either the caller's primary address space or an address or data space on the caller's DU-AL. **Action:** Verify that the parameter list ALET is valid.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM **Meaning:** The version number in the parameter list is not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of z/OS on which your program is running.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. Caller is not enabled. **Action:** Verify that the program is enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Program error. The requestor's primary address space is not equal to the requestor's home address space. **Action:** Ensure that the IXLCSP service is invoked in the correct cross-memory environment.
8	xxxx080D	**Equate Symbol:** IXLRSNCODEAREATOOSMALL **Meaning:** Program error. ANSAREA is too small, as specified by ANSLEN. **Action:** Ensure that the ANSLEN parameter properly reflects the size of the area specified by ANSAREA. Also, make sure that the length of ANSAREA is large enough to contain the data returned.
8	xxxx080E	**Equate Symbol:** IXLRSNCODEBADAREA **Meaning:** Program error. The answer area specified by ANSAREA is either not addressable or not accessible. **Action:** Ensure that the address specified for ANSAREA is valid.
8	xxxx080F	**Equate Symbol:** IXLRSNCODEBADAREAALET **Meaning:** Program error. The ANSAREA ALET is not valid. **Action:** Verify that the ANSAREA ALET is valid.

Table 41. Return and Reason Codes for the IXLCSP Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx081B	**Equate Symbol:** IXLRSNCODENOLENTRIES **Meaning:** Program error. The number of lock table entries specified was 0 for a size computation for a lock structure with record data, or was negative. **Action:** Verify that the number of lock table entries specified is correctly defined.
8	xxxx081C	**Equate Symbol:** IXLRSNCODENOLISTHDRS **Meaning:** Program error. The number of list headers specified on list structure computations must be greater than 0. **Action:** Verify that the number of list headers specified is correctly defined.
8	xxxx081D	**Equate Symbol:** IXLRSNCODEZEROLUSERS **Meaning:** Program error. The number of users specified on lock structure computations must be greater than 0. **Action:** Verify that the number of users specified is correctly defined.
8	xxxx085C	**Equate Symbol:** IXLRSNCODEINVALIDLISTATTR **Meaning:** Program error. List structure computations must specify one of the following: lock entries, data elements, or adjunct. None was specified. **Action:** Verify that a required list structure attribute is specified.
8	xxxx085D	**Equate Symbol:** IXLRSNCODEINVALIDSTGCLASS **Meaning:** Program error. The value of NUMSTGCLASS cannot be 0. **Action:** Verify that you have specified a correct value for NUMSTGCLASS.
8	xxxx085E	**Equate Symbol:** IXLRSNCODEINVALIDCOCLASS **Meaning:** Program error. The value of NUMCOCLASS cannot be 0. **Action:** Verify that you have specified a correct value for NUMCOCLASS.
8	xxxx0860	**Equate Symbol:** IXLRSNCODEDIRRATIO **Meaning:** Program error. The value of DIRRATIO or DIRENTRYCOUNT cannot be 0. Directory entries are required for a cache structure. **Action:** Verify that you have specified a correct value for DIRRATIO or DIRENTRYCOUNT.
8	xxxx0861	**Equate Symbol:** IXLRSNCODEENTRYRATIO **Meaning:** Program error. The value of ENTRYRATIO cannot be 0 when ELEMENTRATIO is greater than 0. ENTRYCOUNT cannot be 0 when either ELEMENTCOUNT is greater than 0 or ADJUNCT=YES. Entries are required for a list structure with data. **Action:** Verify that you have specified a correct value for ENTRYRATIO or ENTRYCOUNT.

Table 41. Return and Reason Codes for the IXLCSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0862	**Equate Symbol:** IXLRSNCODEMAXELEMNUM **Meaning:** Program error. • For a list structure, MAXELEMNUM must be greater than or equal to ELEMENTRATIO divided by ENTRYRATIO if ELEMENTRATIO is not 0. • For a cache structure, MAXELEMNUM must be greater than or equal to ELEMENTRATIO divided by DIRRATIO if ELEMENTRATIO is not 0. The value of MAXELEMNUM cannot be 0. **Action:** Verify that you have specified a correct value for MAXELEMNUM.
8	xxxx086B	**Equate Symbol:** IXLRSNCODEELEMINCRNUM **Meaning:** Program error. The value specified for ELEMINCRNUM is not valid. It must be non-zero and be a power of 2. **Action:** Verify that you have specified a correct value for ELEMINCRNUM.
8	xxxx0871	**Equate Symbol:** IXLRSNCODEMAXELEMNUMELEMCHAR **Meaning:** Program error. The values specified in MAXELEMNUM and either ELEMCHAR or ELEMINCRNUM would result in entries of size greater than 64K. **Action:** Verify that you have specified correct values for MAXELEMNUM and ELEMCHAR or ELEMINCRNUM.

8	xxxx0881	**Equate Symbol:** IXLRSNCODEBADCFLEVEL
		
		Meaning: Program error. Parameters are not appropriate for the CFLEVEL supported by the designated coupling facility. The field at offset SCPA_DiagnosticCode in the answer area designated by ANSAREA further describes the problem.
		
		`Diag        Specified           Minimum`
		`Code        Parameter           CFLEVEL`
		
		`  5      KEYTYPE=SECONDARY         9`
		
		Action: Correct the parameter value in error.

8	xxxx0882	**Equate Symbol:** IXLRSNCODEBADREFOPTION **Meaning:** Program error. The request is rejected because the specified parameter is not appropriate for the value specified for REFOPTION. **For PARAMETER** — **REFOPTION must be** **EMCSTGPCT>0** — KEY **Action:** Correct the parameter value in error.

Table 41. Return and Reason Codes for the IXLCSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0883	**Equate Symbol:** IXLRSNCODEBADEMCSTGPCT **Meaning:** Program error. The value specified for EMCSTGPCT is not valid. The value must be between 0 and 10000. **Action:** Correct the value of EMCSTGPCT.
8	xxxx0888	**Equate Symbol:** IXLRSNCODEBADSTRUCTURESIZE **Meaning:** Program error. The request is rejected because the specified structure size is greater than the specified maximum structure size, or smaller than the calculated marginal structure size. **Action:** Correct the value of STRSIZE.
8	xxxx0889	**Equate Symbol:** IXLRSNCODECALCULATIONOVERFLOW **Meaning:** Program error. The request is rejected because the calculation of the structure size or counts encountered an arithmetic overflow condition. **Action:** Verify that all values specified are correct.
8	xxxx088A	**Equate Symbol:** IXLRSNCODEBADASCMODE **Meaning:** Program error. The request is rejected because the caller is not in primary or AR mode. **Action:** Ensure that the caller is running in primary or AR ASC mode.
8	xxxx088B	**Equate Symbol:** IXLRSNCODEBADELEMCHARORINCRNUM **Meaning:** Program error. The request is rejected because the caller's ELEMCHAR or ELEMINCRNUM specification exceeds the maximum data element size of the input coupling facility. **Action:** Correct the value of either ELEMCHAR or ELEMINCRNUM and resubmit the job.
8	xxxx088C	**Equate Symbol:** IXLRSNCODECOMPUTEREJECTED **Meaning:** Program error. The request is rejected. The request could not be processed due to input that is not valid. **Action:** Examine the CSPA_DiagnosticCode field to identify the input that is not valid.

8	xxxx088E	**Equate Symbol:** IXLRSNCODEINCONSISTENTPARM
		
		Meaning: Program error. Request rejected. A keyword specification was made that also requires one or more other keywords to be specified. The field at offset CSPA_DiagnosticCode in the answer area designated by ANSAREA further describes the problem.
		
		`Diag        Specified             Also`
		`Code        Parameter            Required`
		
		`  1        KEYTYPE=SECONDARY     ADJUNCT=YES`
		
		Action: Correct the parameter value in error.

Table 41. Return and Reason Codes for the IXLCSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. The system from which the request is issued lost connectivity to the coupling facility named by the CFNAME after initial validation of the coupling facility. **Action:** Try the request again after connectivity has been restored, or specify a different coupling facility.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The CFLEVEL of the coupling facility specified by the CFNAME keyword is less than 8. **Action:** Try the request again after specifying a coupling facility of CFLEVEL=8 or higher with the CFNAME keyword.
C	xxxx0C98	**Equate Symbol:** ISLRSNCODECFNOTACCESSIBLE **Meaning:** Environmental error. The coupling facility specified by the CFNAME keyword cannot be accessed from this system. Possible causes include: • The coupling facility is not described by the active CFRM policy. • There is no CFRM couple data set. • The system from which the request is issued does not have connectivity to the coupling facility. • The coupling facility has failed. **Action:** Investigate the possible causes why the coupling facility cannot be accessed, correct the problem, and resubmit the request.
C	FFFFFFFF	**Meaning:** XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES function is not present. **Action:** Re-IPL the system, or follow your particular management protocol.
10	xxxx10xx	**Meaning:** Failure in XES processing. The state of the resource request is unpredictable. **Action:** Save the reason code information and contact the IBM support center.

IXLDISC — Disconnecting from a Coupling Facility Structure

Description

Use the IXLDISC macro to disconnect from a coupling facility structure. You should specify a reason for disconnecting. If the disconnection is normal, specify REASON=NORMAL. If the disconnection is part of an error routine, issue REASON=FAILURE. For disconnections that specify REASON=FAILURE, if the connection disposition is KEEP, the connection is placed in a failed-persistent state after all surviving peer connections acknowledge the disconnect. The connection remains defined to the coupling facility.

In order to disconnect from a structure, the requestor must specify the CONTOKEN from the IXLCONN request issued for the structure. The IXLDISC macro must also be issued from the same task that issued the IXLCONN macro for the structure. For more information on IXLDISC, see *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN. The primary address space must be equal to the requestor's primary address space at the time of the connection.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Restrictions

The IXLDISC macro must be issued from the same task that issued the IXLCONN macro for the structure.

If disconnecting during the rebuild process, issue IXLDISC under the same task that issued the original IXLCONN request and specify the original contoken.

Note that if you disconnect from a lock structure with locks held, the system treats it as disconnecting with REASON=FAILURE.

Input Register Information

Before issuing the IXLDISC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLDISC macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if applicable, if GPR 15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLDISC macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLDISC. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Performance Implications

After IXLDISC is issued, all peer connections are notified of the disconnect/failure event. The performance of IXLDISC depends on recovery and cleanup protocols of the surviving connections.

IXLDISC processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLDISC Version Support

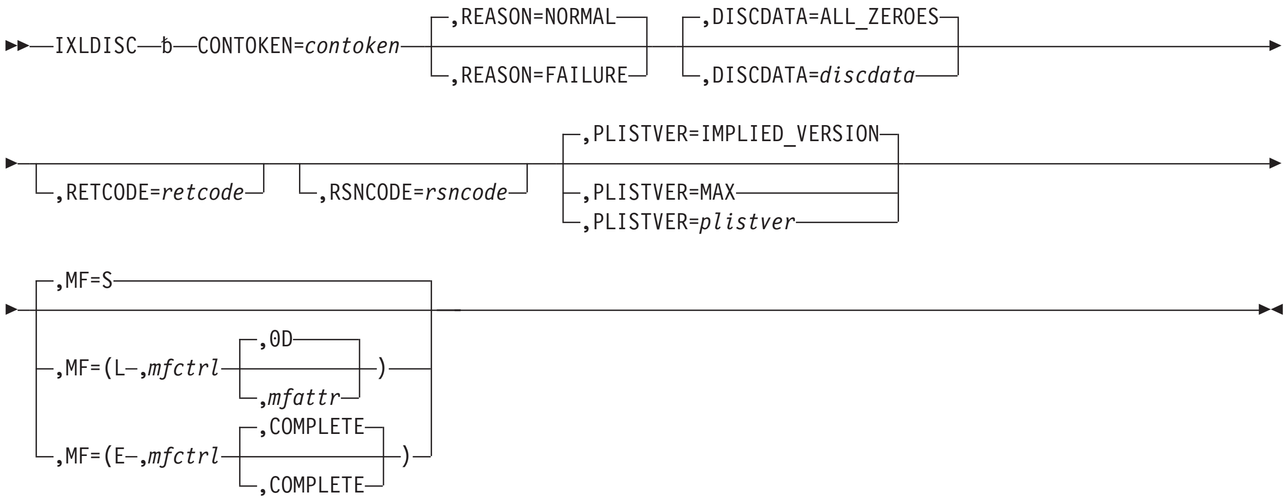
The IXLDISC macro supports version 0 keywords and functions.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLDISC macro is as follows:

IXLDISC diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

CONTOKEN=*contoken*
Use this input parameter to specify the connection identifier that was returned by the IXLCONN service. The connection identifier uniquely identifies the user's connection to the structure.

During rebuild processing, use the original, not the temporary, contoken.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the connect token.

,DISCDATA=ALL_ZEROES
,DISCDATA=*discdata*
Use this input parameter to specify eight bytes of connector data that the system will pass to the event exit of other connectors when you disconnect from the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the eight-character field.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX

,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLDISC Version Support” on page 634 for a description of the options available with PLISTVER.

,REASON=NORMAL
,REASON=FAILURE
Use this input parameter to specify the reason for the disconnection: The disconnect reason is presented to all surviving connectors’ event exits.

NORMAL
Indicates that the requestor normally disconnects from the structure. The disposition for the connection specified on the IXLCONN macro (CONDISP) does not apply and the connection is made not-defined.

If a connection to a lock structure owning the locking resources issues the IXLDISC REASON=NORMAL, the disconnect is treated as if the user specified REASON=FAILURE.

FAILURE
Indicates a disconnection that is the result of an error or abnormal end. If the connection fails with REASON=FAILURE, the connector’s CONDISP applies. After all surviving connections are notified and confirm the disconnect, the connection is made not-defined (CONDISP=DELETE) or failed-persistent (CONDISP=KEEP).

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

Return and Reason Codes

When the IXLDISC macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, if applicable, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 42. Return and Reason Codes for the IXLDISC Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None	**Meaning:** Disconnection is successful. The connect token is now invalid and will be rejected on any subsequent requests. **Action:** None.
4	xxxx0401	**Equate Symbol:** IXLRSNCODEOWNINGRESOURCES **Meaning:** Disconnection is successful. Requestor owned the resources in the lock structure from which the requestor has disconnected. The disconnect will be treated as abnormal regardless of whether the connector specified REASON=NORMAL. If the connection is CONDISP=KEEP, the connector is made failed-persistent after all confirmations are received. **Action:** The system invalidates the connect token of the requestor.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The IXLDISC parameter list is not accessible. **Action:** Verify that: • The parameter list address is uncorrupted • The parameter list is addressable in the caller's primary address space. • If you are calling IXLDISC in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLDISC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLDISC.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** Program error. The IXLDISC parameter list ALET is not valid. **Action:** Verify that: • If you are calling IXLDISC in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLDISC in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing the IXLDISC. • If you are not running in AR-mode, and the parameter list does not need to be ALET-qualified, the corresponding access register should contain zeros.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** Program error. The IXLDISC parameter list contains an invalid version number **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS your program is running on.
8	xxxx0805	**Equate Symbol:** IXLRSNCODEBADTCB **Meaning:** Program error. The task that issued the IXLDISC macro is different from the task that issued the corresponding IXLCONN macro. **Action:** Set up your program so that it does the disconnect under the same TCB that the connect was done under.

Table 42. Return and Reason Codes for the IXLDISC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** Program error. The requestor is in SRB mode. **Action:** You must be running in task mode under the task that did the connect to issue IXLDISC.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. The requestor is not in an enabled state. **Action:** The requestor must be enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Program error. The requestor's primary address space is not the same as the home address space. **Action:** The primary address space must equal the home address space to issue the IXLDISC. The primary address space must be the same address space the IXLCONN was issued from.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The requestor specified a CONTOKEN that is not valid, or the request was issued from an address space other than the connector's address space. **Action:** Verify the CONTOKEN specified was the original one received in the IXLCONN answer area after the connect was issued. The disconnect must be issued from the same address space as the connect.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. There are no coupling facility services available. The hardware support necessary to provide coupling facility services might not be present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	—	**Meaning:** XES processing has failed. **Action:** Save the reason code information and contact the IBM support center.

IXLEERSP — Responding to an Event

Description

Use the IXLEERSP macro if you are an active connection to a structure in the coupling facility and must provide an asynchronous response to an event exit event (at a time other than when the event exit gets control). Use IXLEERSP to respond to the following specific events reported to your event exit:

- Disconnected or Failed Connection event
- Existing Connection event for a failed-persistent connection
- Rebuild Quiesce event to start rebuilding or duplexing a structure
- Rebuild Cleanup event to indicate that resource clean-up for rebuilding or duplexing a structure is complete
- Rebuild Stop event to confirm that the system stop the rebuilding or duplexing process
- Rebuild Connection Failure event to confirm the failure of a rebuild connection
- Structure Temporarily Unavailable event to acknowledge that a system-managed process has been initiated.

For EVENT=DISCFAILCONN, the requestor is able to indicate, using RELEASECONN=YES, that all needed recovery for the failed connector has been performed, and that, as a result, the failed connector need not be made failed-persistent when all confirmations are received. Using PROXYRESPONSE=YES, the requestor is also able to provide a response on behalf of a peer connector that is the subject of a Disconnected or Failed Connection event. The events for which the response can be provided are the Rebuild Cleanup and Rebuild Stop events.

Before using IXLEERSP, the requestor must have set a return code X'08' in the event exit parameter list (IXLYEEPL). See field IXLRCEVENTEXITLATERESPONSE in IXLYCON. Setting the return code indicates, at the time the event is presented to the event exit, that the requestor intends to issue IXLEERSP to respond to the event. If the connector intends to respond to the event asynchronously, the connector must use the IXLEERSP macro.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN, The primary address space must be equal to the requestor's primary address space at the time of the connection.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Restrictions

The connector must set return code X'08' in IXLYEEPL within the event exit to indicate that IXLEERSP is to provide the response to the event.

Input Register Information

Before issuing the IXLEERSP macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLEERSP macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if applicable, if GPR 15 return code is non-zero.
1	Used as work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller of the IXLEERSP macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLEERSP. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Performance Implications

IXLEERSP processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLEERSP Version Support

The IXLEERSP macro supports versions in the range of 0 - 2.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXLEERSP macro.
- The following keyword is supported by version 1 and subsequent versions of the IXLEERSP macro.

PROXYRESPONSE

- The following event type is supported by version 2 and subsequent versions of the IXLEERSP macro.

STRTEMPUNAVAIL

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLEERSP macro is written as follows:

main diagram

```
►►──IXLEERSP──b──EVENT=──┬─DISCFAILCONN──┤ parameters-1 ├──────────────────────────┬──────────────────►
                         ├─EXISTINGCONN───,SUBJCONTOKEN=subjcontoken────────────────┤
                         ├─STRTEMPUNAVAIL───,EVENTSEQ#=eventseq#────────────────────┤
                         ├─REBLDQUIESCE─────────────────────────────────────────────┤
                         ├─REBLDSTOP──┤ parameters-2 ├──────────────────────────────┤
                         ├─REBLDCLEANUP──┤ parameters-2 ├───────────────────────────┤
                         └─REBLDCONNFAIL─,SUBJCONTOKEN=subjcontoken─,EVENTSEQ#=eventseq#─┘


                                                                     ┌─,MF=S────────────────────────────┐
►─,CONTOKEN=contoken──┬───────────────────┬──┬───────────────────┬──┼──────────────────────────────────┼──►◄
                      └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘  │                ┌─,0D──────┐      │
                                                                     ├─,MF=(L─,mfctrl─┼──────────┼─)────┤
                                                                     │                └─,mfattr──┘      │
                                                                     │                ┌─,COMPLETE─┐     │
                                                                     └─,MF=(E─,mfctrl─┼───────────┼─)───┘
                                                                                      └─,COMPLETE─┘
```

parameters-1

```
►►──,SUBJCONTOKEN=subjcontoken──,EVENTSEQ#=eventseq#──┬─,RELEASECONN=NO──┬──────────────────────►◄
                                                      └─,RELEASECONN=YES─┘
```

parameters-2

```
     ┌─,PROXYRESPONSE=NO──────────────────────────────────────┐
►►───┼────────────────────────────────────────────────────────┼──────────────────────────────►◄
     └─,PROXYRESPONSE=YES───,SUBJCONTOKEN=subjcontoken────────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined.

,CONTOKEN=*contoken*

Use this input parameter to specify the responding connection's connect token. CONTOKEN is returned to the connect answer area when the connector issues the IXLCONN macro for the structure.

Note that during rebuild, you must specify the original contoken, not the temporary one.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the requestor's connect token.

EVENT=DISCFAILCONN
EVENT=EXISTINGCONN
EVENT=REBLDQUIESCE
EVENT=REBLDSTOP
EVENT=REBLDCLEANUP
EVENT=REBLDCONNFAIL
EVENT=STRTEMPUNAVAIL

Use this input parameter to specify the connection event to which the requestor is responding:

DISCFAILCONN

Indicates the Disconnected or Failed Connection event (equal to field EEPLDISCFAILCONNECTION in the IXLYEEPL parameter list). The SUBJCONTOKEN keyword is required to identify the connection that is the subject of the event. The RELEASECONN keyword is required. It indicates whether the connection should remain persistent or be released.

EXISTINGCONN
Indicates an Existing Connection event for a connection that is failed-persistent (equal to field EEPLEXISTINGCONNECTION in the IXLYEEPL parameter list). This event is reported to new connections only. Specifying EVENT=EXISTINGCONN deletes the failed-persistent connection from the structure, and the existing connection is no longer defined to the coupling facility. The SUBJCONTOKEN keyword is required to identify the connection that is the subject of the event.

REBLDQUIESCE
Indicates the Rebuild Quiesce event to rebuild or duplex a structure (equal to field EEPLREBUILDQUIESCE in the IXLYEEPL parameter list). Rebuild is initiated through the SETXCF START,REBUILD command or the IXLREBLD REQUEST=START request. Duplexing is initiated through the SETXCF START,REBUILD,DUPLEX command or the IXLREBLD REQUEST=STARTDUPLEX request. (See "IXLREBLD — Rebuilding or Duplexing a Structure" on page 1171.) Users must respond to the Rebuild Quiesce event. Specifying EVENT=REBLDQUIESCE indicates that the requestor participating in the user-managed process to rebuild or duplex the structure has quiesced all activity to the structure including the completion of processing based on the use of restart tokens.

Note that if you do not provide a REBLDQUIESCE response, you must either stop the rebuild or duplex process or disconnect from the structure.

REBLDSTOP
Indicates the Rebuild Stop event to stop rebuilding or duplexing a structure (equal to field EEPLREBUILDSTOP in the IXLYEEPL parameter list). Stop rebuilding is initiated through the SETXCF STOP,REBUILD command or the IXLREBLD REQUEST=STOP request. Stop duplexing is initiated through the SETXCF STOP,REBUILD,DUPLEX command or the IXLREBLD REQUEST=STOPDUPLEX request. (See "IXLREBLD — Rebuilding or Duplexing a Structure" on page 1171.) Users must respond to the Rebuild Stop event. Specifying EVENT=REBLDSTOP indicates that the requestor is participating in stopping the rebuild or duplexing process.

REBLDCLEANUP
Indicates the Rebuild Cleanup event (equal to field EEPLREBUILDCLEANUP in the IXLYEEPL parameter list). When all connectors to a rebuilt structure issue IXLREBLD REQUEST=COMPLETE to indicate that they have completed processing, the system reports the Rebuild Cleanup event to the event exit. When all connectors to a duplexed structure issue IXLREBLD REQUEST=DUPLEXCOMPLETE to indicate that they have completed processing, the system also reports the Rebuild Cleanup event to the event exit. Users must respond to this event to confirm that resource cleanup for rebuilding is complete. Specifying EVENT=REBLDCLEANUP indicates that the requestor has completed the rebuilding process and has no knowledge of the old structure or has completed the duplexing process and has no knowledge of the old structure.

REBLDCONNFAIL
Indicates that the connection identified by SUBJCONTOKEN failed in its connection to the new structure during rebuild processing. The SUBJCONTOKEN keyword is required to identify the connection that is the subject of the event.

STRTEMPUNAVAIL
Indicates that the structure is temporarily unavailable for coupling facility requests against it. The structure is undergoing a system-managed process, such as system-managed rebuild. Specifying EVENT=STRTEMPUNAVAIL indicates that the responder has completed preparations for the structure being unavailable for the duration of the system-managed process. The EVENTSEQ# is required to identify the specific Structure Temporarily Unavailable event for which the response applies.

,EVENTSEQ#=*eventseq#*
Use this input parameter to specify the event sequence number. The event sequence number for the event must be the same value that was presented to this connection in the event exit parameter list (IXLYEEPL).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the event sequence number.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PROXYRESPONSE=NO
,PROXYRESPONSE=YES

Use this input parameter to indicate whether this response is being provided on behalf of a failing connector.

NO

Indicates that this is not a proxy response. The response is on behalf of the connector described by the CONTOKEN keyword.

YES

Indicates that this is a proxy response. The response is being provided on behalf of the connector described by the SUBJCONTOKEN keyword.

Note: To ensure that the PROXYRESPONSE feature is installed on the system on which you are running, issue IXCQUERY REQINFO=FEATURES. QUREQRFPROXYRESPONSE, returned from the IXCQUERY request, indicates whether the PROXYRESPONSE feature is installed.

,RELEASECONN=NO
,RELEASECONN=YES
Use this input parameter to specify whether the connection should remain persistent or be released.

NO
Indicates that the requestor has completed processing for the failed connection event, and specifies that XES should continue processing for the failed connection. The persistence attribute of the connection is not affected by this response.

YES
Indicates that the requestor has completed processing for the failed connection event, specifies that XES should continue processing for the failed connection, and specifies that this connection is no longer required to be persistent.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,SUBJCONTOKEN=*subjcontoken*
Use this input parameter to specify the CONTOKEN for the connection described by the event. SUBJCONTOKEN must be the same value that was presented to the event exit in field EEPLSUBJCONTOKEN of the IXLYEEPL.

Do not provide a proxy response on behalf of a failing connector until AFTER the EEPLDISCFAILCONNECTION event for the subject user has been presented to your event exit. You can use the value of the EEPLSUBJCONTOKEN field from that invocation of the event exit as input for this parameter.

Note that if the subject connector is not in the failing state, the IXLEERSP request will fail with the IXLRSNCODESUBJCONNNOTFAILING reason code.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the CONTOKEN for the connection described by the event.

Return and Reason Codes

When the IXLEERSP macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 43. Return and Reason Codes for the IXLEERSP Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None	**Meaning:** Processing for IXLEERSP has completed successfully. **Action:** None.
4	xxxx041D	**Equate Symbol:** IXLRSNCODEIGNOREFORREBUILDSTOP **Meaning:** The response for the Rebuild Quiesce event was ignored because a Rebuild Stop is in progress. **Action:** You should have been notified of the Rebuild Stop event. You may have issued IXLEERSP for the wrong event. You may have issued IXLEERSP for the Rebuild Quiesce event more than once. You may have connected during the Rebuild Quiesce window and a Rebuild Stop has been processed already. If so, respond to the Rebuild Stop event. Check your protocol for the rebuild process.
4	xxxx0428	**Equate Symbol:** IXLRSNCODEIGNOREFORSYSMGDSTOP **Meaning:** The response for the Structure Temporarily Unavailable event was ignored because the system-managed rebuild process has been stopped. **Action:** None.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** Program error. The IXLEERSP parameter list ALET is not valid. **Action:** All parameters must be in the primary address space so all ALETs should be set up accordingly. Fix the ALET of the parameter list and re-issue the macro.

Table 43. Return and Reason Codes for the IXLEERSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** Program error. The requestor is in SRB mode. **Action:** You must be running in task mode to issue this macro.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. The requestor is not in an enabled state. **Action:** The user must be enabled for I/O and external interrupts to issue this macro.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Program error. The primary address space does not equal the home address space. **Action:** The primary address space must be equal to the primary address space when the IXLCONN was issued. The home address space must be equal to the primary address space.

Table 43. Return and Reason Codes for the IXLEERSP Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C3F	**Equate Symbol:** IXLRSNCODECONNNOTDEFINED **Meaning:** Environmental error. The CONTOKEN is for a connection that is in the not defined state. **Action:** The connection issuing the IXLEERSP macro must be active. Verify the CONTOKEN passed. Check your protocol to determine why you are responding to this event.

Table 43. Return and Reason Codes for the IXLEERSP Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C40	**Equate Symbol:** IXLRSNCODECONNNOTACTIVE **Meaning:** Environmental error. The CONTOKEN is for a connection that is not in the active state. **Action:** The connection issuing the IXLEERSP macro must be in the active state. Verify the CONTOKEN passed. Check your protocol to determine why you are responding to this event.
C	xxxx0C41	**Equate Symbol:** IXLRSNCODEUNEXPECTEDRESPONSE **Meaning:** Environmental error. This connection is responding to an event via IXLEERSP to which the system is not currently expecting a response. **Action:** Prior to issuing the IXLEERSP macro, an event exit must have passed a return code of X'8' in the IXLYEEPL to indicate to XES that a response will be made to the event through the IXLEERSP macro. Otherwise, the response will not be expected. Check your event exit, and your protocol for handling events.
C	xxxx0C42	**Equate Symbol:** IXLRSNCODEINVALIDEVENT **Meaning:** Environmental error. The response provided via IXLEERSP is not one that is expected for the event presented. **Action:** Verify the event specified and the parameters passed on the IXLEERSP.
C	xxxx0C48	**Equate Symbol:** IXLRSNCODESUBJCONNNOTDEFINED **Meaning:** Environmental error. The connection identified by SUBJCONTOKEN is in the not defined state. **Action:** Verify that your parameter list has not been overlaid. Verify that SUBJCONTOKEN is from EEPLSUBJCONTOKEN from the EEPL passed to the event exit.
C	xxxx0C49	**Equate Symbol:** IXLRSNCODEREBUILDEERSPIGNORED **Meaning:** Environmental error. EVENT=REBLDCONNFAIL is not valid because the connection identified by SUBJCONTOKEN is not in the active state. The original connection has terminated. **Action:** None necessary, but check your protocol to determine why you thought you needed to respond to this event. Verify that SUBJCONTOKEN is from EEPLSUBJCONTOKEN from the EEPL passed to the event exit. You may have responded more than once to this event.
C	xxxx0C6D	**Equate Symbol:** IXLRSNCODESUBJCONNNOTFAILING **Meaning:** Environmental error. An attempt to respond by proxy on behalf of the connector identified by the value provided for the SUBJTOKEN keyword failed because that connector is not in the failing state. **Action:** Ensure that you do not respond by proxy on behalf of a failing connector until after you have been presented with the EEPLDISCFAILCONNECTION event exit.

Table 43. Return and Reason Codes for the IXLEERSP Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C91	**Equate Symbol:** IXLRSNCODESYSMGDRESPONSENOTPERMITTED **Meaning:** The structure is in system-managed processing. A response is not permitted from the connection. The request is not processed. Applies to the following events: REBLDQUIESCE, REBLDCONNFAIL, REBLDCLEANUP, and REBLDSTOP. **Action:** Check your protocol for system-managed processes.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** There are no coupling facility services available. The hardware support necessary to support coupling facility services might not be present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	None	**Meaning:** XES processing has failed. **Action:** Save the reason code information and contact the IBM support center.

IXLFCOMP — Wait for Completion or Obtain Status of an IXLLIST or IXLCACHE Request

Description

If you are an IXLLIST or IXLCACHE macro user who specified MODE=ASYNCTOKEN or specified MODE=SYNCTOKEN but had the request processed asynchronously, you can use the IXLFCOMP macro to do either of the following:

- Test whether your list or cache request has completed (OPTYPE=TEST). Choose this option if your task cannot be suspended or your program can perform other work while the list or cache request is being processed. IXLFCOMP's return code indicates whether the request has completed.
- Have your task suspended until the list or cache request completes (OPTYPE=COMPLETE). Choose this option if your task can be suspended and you have no other work to perform while the list or cache request is being processed.

  When the request has completed (if it has not completed when you issue IXLFCOMP), control returns to you so you can check the results of the request in the output areas you specified on the list or cache request.

Note: The following information assumes that you are familiar with either the IXLLIST macro or the IXLCACHE macro, and that you are using either a list or cache structure.

For More Information

The IXLFCOMP guidance information in *z/OS MVS Programming: Sysplex Services Guide* contains additional information about using the IXLFCOMP macro, including the use of IXLFCOMP in recovery scenarios.

For more information about the IXLLIST macro, see "IXLLIST — List Services" on page 665.

For more information about the IXLCACHE macro, see "IXLCACHE — Cache Services" on page 245.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	HASN=PASN, any SASN. The primary address space must equal the primary address space of the caller of the IXLCONN macro.
AMODE:	31-bit
ASC mode:	Primary or AR
Interrupt status:	If you specify OPTYPE=TEST, your program might be enabled or disabled for I/O and external interrupts. If you specify OPTYPE=COMPLETE, your program must be enabled (so that it can be suspended if necessary).
Locks:	If your program is disabled, it must hold the CPU lock. Otherwise, no locks may be held.
Control parameters:	See "Restrictions" on page 652

Programming Requirements

- If your program is in AR mode, issue SYSSTATE ASCENV=AR before you issue the IXLFCOMP macro. ASCENV=AR causes the system to generate code appropriate for AR mode. For more information about the SYSSTATE macro, see *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.
- Include the IXLYCON mapping macro to generate the equate symbols for the return codes.

Restrictions

- You can issue IXLFCOMP requests only for IXLLIST or IXLCACHE operations initiated by your own connection.
- Only one IXLFCOMP request can run at a time with a particular request token. If you attempt to issue another IXLFCOMP request with the same token while the first is running, your second request fails.
- All virtual storage areas passed to the IXLFCOMP macro must be addressable in at least one of the following ways:
  - From the primary, home, or secondary address space
  - From the PASN access list
  - From the DU access list
- Any virtual storage area specified on the corresponding prior IXLLIST or IXLCACHE request must still be addressable as it was at the time of the prior request. See the IXLLIST or IXLCACHE macros for the addressability requirements of the virtual storage areas associated with those requests.
- If you are running disabled, the parameter list and all storage areas addressed by macro parameters must reside in either fixed or disabled reference (DREF) storage. Furthermore, any virtual storage areas that were specified on the corresponding prior IXLLIST or IXLCACHE invocation must also reside in fixed or DREF storage if the IXLFCOMP caller is disabled.
- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).

Input Register Information

Before issuing the IXLFCOMP macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code, if applicable
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

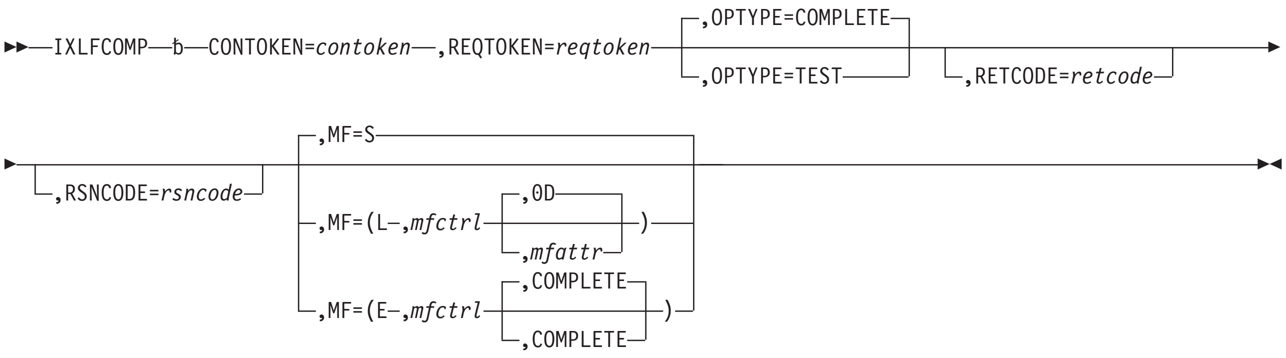
Performance Implications

If OPTYPE=COMPLETE is specified, your task is suspended until the IXLLIST or IXLCACHE request completes.

Syntax Diagram

The syntax of the IXLFCOMP macro is as follows:

IXLFCOMP diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

CONTOKEN=*contoken*
Use this input parameter to specify the connect token you received when you issued the IXLCONN macro to connect to the cache or list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,OPTYPE=COMPLETE
,OPTYPE=TEST
Use OPTYPE=COMPLETE to request that the IXLFCOMP request return information only when the list or cache request identified by REQTOKEN has completed. If the request identified by REQTOKEN has not completed, your task will be suspended until it has completed.

Use OPTYPE=TEST to test the current processing status of the list or cache request. If your task cannot be suspended, use this option to poll for request completion.

,REQTOKEN=*reqtoken*
Use this input parameter to specify the asynchronous request token returned by the IXLLIST or IXLCACHE request. This token identifies the asynchronous request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the asynchronous request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the IXLFCOMP macro returns control to your program:
- GPR 15 (and *retcode* if you coded RETCODE) contains a return code.
- If the return code is not zero, GPR 0 (and *rsncode* if you coded RSNCODE) contains a reason code, if applicable.

You could receive any of the return codes and reason codes issued by IXLFCOMP, in addition to those described for IXLLIST and IXLCACHE. The return and reason codes that pertain to the IXLFCOMP request are listed in Table 44 on page 655. Otherwise, the return and reason codes pertain to the prior IXLLIST and IXLCACHE request that has completed. Refer to those services for the return and reason code information.

The answer area is not valid if you issue an IXLFCOMP OPTYPE=COMPLETE request and get back RC=IXLRETCODEPARMERROR with RSN=IXLRSNCODENOSUSPENDISABLE.

The equate symbols associated with each IXLFCOMP return code are as follows:

4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR

For the reason codes shown below, *xxxx* denotes information that is not part of the reason code.

Table 44. Return and Reason Codes for the IXLFCOMP Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0413	**Equate symbol:** IXLRSNCODEREQNOTCOMP **Meaning:** The request identified by REQTOKEN has not completed. The IXLLIST or IXLCACHE answer area is not valid. **Action:** Issue the IXLFCOMP macro again, with OPTYPE=TEST to determine when the list or cache request will complete (with the answer area filled in) or with OPTYPE=COMPLETE to wait for the list or cache request to complete.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 44. Return and Reason Codes for the IXLFCOMP Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0831	**Equate symbol:** IXLRSNCODEBADREQTOKEN **Meaning:** Program error. You specified an asynchronous request token that was not valid. Your IXLFCOMP request failed. The IXLLIST or IXLCACHE answer area is not valid. **Action:** Ensure that the request token passed on the IXLFCOMP request is the same one that you got back from IXLLIST or IXLCACHE on a SYNCTOKEN or ASYNCTOKEN request. Verify that if the IXLLIST or IXLCACHE request was a SYNCTOKEN request, that it was in fact processed asynchronously. If the request was processed synchronously, the request token is not meaningful and there is no reason to use the IXLFCOMP macro. Verify that you are not re-issuing the IXLFCOMP macro using the same request token after observing the completion of the request. Once you observe completion of a request, the request token is no longer valid, and any subsequent IXLFCOMP requests will complete with this reason code.
8	xxxx0851	**Equate symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. A request specifying OPTYPE=COMPLETE failed because the prior IXLLIST or IXLCACHE request has not completed, and the caller is disabled and cannot be suspended. **Action:** Release the CPU lock to become enabled and reissue the request.
10	xxxx10xx	**Equate symbol:** IXLRETCODECOMPERROR **Meaning:** XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. **Action:** Contact the IBM Support Center.

IXLFORCE — Deleting a Persistent Structure or Failed Connection

Description

The IXLFORCE service is intended only for use as a cleanup utility; therefore, you do not need to be connected to a structure to invoke the macro. IXLFORCE allows the requestor to delete:
- A persistent structure
- A failed-persistent connection to a structure
- All failed-persistent connections to a structure
- A structure dump
- The serialization held on a structure for a structure dump

A persistent structure can be deleted if :
- There are no longer any active or failed-persistent connections to the structure.
- There are no failed-persistent connections pending reconciliation into the CFRM active policy.
- There is no structure dump associated with the structure.

An active connection cannot be deleted. A failed-persistent connection can be deleted only if:
- All connectors active at the time the failed-persistent connection terminated have either provided an event exit response acknowledging the termination of the failed-persistent connector, or have themselves been terminated.

While structure rebuild is in progress:
- A structure cannot be deleted because a structure that has a status of rebuild in progress must have at least one active connector.
- A failed-persistent connection cannot be deleted.

While structure duplexing is in progress, a persistent structure cannot be deleted nor can a failed-persistent connection be deleted except as noted below.
- When a persistent structure is in the Duplex Established phase of a duplexing rebuild and a request to stop the duplexing and switch to the new structure is not in progress, the structure can be deleted. Both the old and the new structure are deleted.
- When a failed-persistent connection to a structure that is in the Duplex Established phase of duplexing rebuild and a request to stop the duplexing and switch to the new structure is not in progress, the failed-persistent connection can be deleted. Failed-persistent connectors to both instances of the structure will be deleted.

Note that forcing the deletion of a structure or a connection without understanding how the structure is being used may cause loss of data.

The security administrator may have controlled the use of installation structures through the use of RACF or another security product. If so, ensure that you are authorized to issue the IXLFORCE macro for the structure.

For more information about structure and connection persistence, see *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7

Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Restrictions

- The requestor can delete a persistent structure only if there are no active or failed-persistent connections to the structure, there are no failed-persistent connections pending reconciliation into the CFRM active policy, and there is no structure dump associated with the structure.

  A structure cannot be deleted while structure rebuild is in progress for the structure, because a structure for which rebuild is in progress must have at least one active connector.

  A structure in the duplexing rebuild process can be deleted only if there are no active or failed-persistent connections, the structure is in the Duplex Established phase, and a switch to the new structure is not in progress.
- The requestor can delete a failed-persistent connection. The failed-persistent connection will be deleted from the old structure, the new structure, or both instances of the structure. However, an active connection cannot be deleted, nor can a failed-persistent connection be deleted while structure rebuild is in progress for the structure.

  A connection to a structure in the duplexing rebuild process can be deleted only when the connection is failed-persistent and the structure is in the Duplex Established phase with a switch to the new structure not in progress.
- The requestor can delete only a single structure, structure dump, or structure dump serialization with one invocation of the IXLFORCE macro. However, the requestor can delete multiple failed-persistent connections to a structure with one invocation of the IXLFORCE macro.

  A program can use the IXCQUERY macro to determine the set of objects that are candidates for deletion. In the case of forcing all failed-persistent connections to a structure, IXCQUERY can be used to determine the set of connections that either will be or were forced.

Input Register Information

Before issuing the IXLFORCE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLFORCE macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code if applicable
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLFORCE macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLFORCE. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Performance Implications

IXLFORCE processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Syntax Diagram

The syntax of the IXLFORCE macro is as follows:

IXLFORCE diagram

```
►►──IXLFORCE──b──REQUEST=──┬─STRUCTURE─,STRNAME=strname──────────────────────┬────────────────────►
                           ├─CONNECTION─,STRNAME=strname─,CONNAME=conname────┤
                           ├─ALLCONNS─,STRNAME=strname───────────────────────┤
                           ├─STRDUMP─,STRNAME=strname─┬────────────────────┬─┤
                           │                          └─,STRDUMPID=strdumpid┘ │
                           └─STRDUMPSERIAL─,STRNAME=strname─┬────────────────────┬┘
                                                            └─,STRDUMPID=strdumpid┘


                                                   ┌─,MF=S─────────────────────────────┐
►─┬──────────────────┬─┬──────────────────┬─┼───────────────────────────────────┼──────────────►◄
  └─,RETCODE=retcode─┘ └─,RSNCODE=rsncode─┘ │                ┌─,0D──────┐        │
                                            ├─,MF=(L─,mfctrl─┼──────────┼─)──────┤
                                            │                └─,mfattr──┘        │
                                            │                ┌─,COMPLETE─┐       │
                                            └─,MF=(E─,mfctrl─┼───────────┼─)─────┘
                                                             └─,COMPLETE─┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,CONNAME=*conname*

Use this input parameter to specify the connect name of the connection to be deleted. The connection identified is connected to the structure specified by STRNAME.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the connect name of the connection to be deleted.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

REQUEST=STRUCTURE
REQUEST=CONNECTION
REQUEST=ALLCONNS
REQUEST=STRDUMP
REQUEST=STRDUMPSERIAL
Use this input parameter to specify what the requestor wants to delete.

STRUCTURE
Indicates that the structure is to be deleted. A structure can be deleted only when there are no active or failed-persistent connections to the structure. You can use IXCQUERY to determine the state of connections to a structure. If there is a structure dump associated with the structure, deallocation of the structure remains pending until the structure dump is released.

CONNECTION
Indicates that the specified failed-persistent connection to a structure is to be deleted. If the last connection to a non-persistent structure is deleted, the structure also is deleted.

ALLCONNS
Indicates that all failed-persistent connections to a structure are to be deleted. If the last connection to a non-persistent structure is deleted, the structure also is deleted.

STRDUMP
Indicates that a structure dump associated with either an active structure or a structure pending deallocation is to be deleted. Structures pending deallocation will be processed only if a nonzero STRDUMPID is specified.

STRDUMPSERIAL
Indicates that the serialization held on the structure for the structure dump is to be released. For

structures pending deallocation, however, release of dumping serialization is not supported because such structures have no active connectors who might be impacted by the dump serialization.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STRDUMPID=0
,STRDUMPID=*strdumpid*

Use this input parameter to specify the structure dump identifier of the structure dump to be deleted. or of the dump for which structure dump serialization is to be released. This is used to uniquely identify the specific instance of a structure dump, since there might be multiple instances of a structure, each with an associated structure dump.

If you do not specify STRDUMPID or specify a value of zero on STRDUMPID, the system deletes, or releases dump serialization for, all structure dumps associated with the allocated structure. To delete a structure dump for a structure pending deallocation, you must specify a non-zero STRDUMPID. Release of structure dump serialization is not supported for a structure pending deallocation because such structures have no active connectors who might be impacted by the dump serialization. If a structure rebuild is in process, the system might delete, or release dump serialization for, the dump associated with the rebuild old structure, rebuild new structure, or both.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword input field that contains the structure dump identifier of the structure dump to be deleted.

,STRNAME=*strname*

Use this input parameter to specify the name of the structure that is identified on the IXLFORCE request. The structure might be active or pending deallocation. You can use the IXCQUERY macro to determine which structures have been defined in the CFRM active policy.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the structure name.

Return and Reason Codes

When the IXLFORCE macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 45. Return and Reason Codes for the IXLFORCE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None	**Meaning:** IXLFORCE processing completed successfully. The structure, failed-persistent connection, all failed-persistent connections, structure dump, or structure dump serialization has been deleted as requested. **Action:** None.
4	xxxx041A	**Equate Symbol:** IXLRSNCODEFORCECONNDELSTR **Meaning:** The IXLFORCE request to delete a connection or all connections succeeded, but the structure disposition was DELETE, and no active connections to the structure remained. The structure was deallocated. **Action:** None necessary. However, if this result is unexpected, determine if the structure disposition was specified properly in IXLCONN.
4	xxxx041C	**Equate Symbol:** IXLRSNCODEPENDING **Meaning:** The IXLFORCE request could not be processed immediately. The request is pending, and the system will process it when the condition causing the delay is resolved. **Action:** None necessary. If the delay is due to a dump associated with the structure, you might supply a dump data set for SDUMPX to write the dump or you might delete the dump.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The IXLFORCE parameter list is not accessible. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are invoking IXLFORCE in AR-mode and you specified the parameter list address using implicit register notation, the corresponding access register was updated appropriately. • If you are invoking IXLFORCE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLFORCE.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** The IXLFORCE parameter list ALET is not valid. **Action:** Ensure that the ALET is zero or that the ALET represents a valid entry on the DU-AL.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** Version number in the IXLFORCE parameter list is not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS that your program is running on.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** The requestor is in SRB mode. The request fails. **Action:** Do not issue the IXLFORCE macro when running in SRB mode.

Table 45. Return and Reason Codes for the IXLFORCE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** The requestor is not in an enabled state. The request fails. **Action:** Ensure that the user is enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** Issuer's primary address space is not equal to the home address space. **Action:** Make sure that the primary address space and the home address space are the same at the time of the IXLFORCE invocation.
8	xxxx084C	**Equate Symbol:** IXLRSNCODEBADNOSAFAUTH **Meaning:** Environmental error. The user does not have the proper SAF authorization. The request fails. **Action:** Determine why the installation has not allowed the proper SAF authorization for access to the structure.
C	xxxx0C05	**Equate Symbol:** IXLRSNCODESTRNOTINPOLICY **Meaning:** Environmental error. The requested structure is not in the CFRM active policy. **Action:** Specify a structure that is currently defined in the CFRM active policy or switch to a new CFRM policy that contains the structure for which this IXLFORCE was invoked.
C	xxxx0C0A	**Equate Symbol:** IXLRSNCODESTRNOTALLOCATED **Meaning:** Environmental error. The requested structure is not currently allocated. **Action:** Ensure that the structure name has been specified correctly.
C	xxxx0C26	**Equate Symbol:** IXLRSNCODECONACTIVE **Meaning:** Environmental error. The requested connection is active. IXLFORCE can only delete connections in a failed-persistent state. **Action:** Ensure that the connection to be deleted is in a failed-persistent state.
C	xxxx0C27	**Equate Symbol:** IXLRSNCODERSPNOTREC **Meaning:** Environmental error. One or more surviving connections have not provided an event exit response for the requested connection. **Action:** Examine your protocol to determine why responses have not been provided.
C	xxxx0C28	**Equate Symbol:** IXLRSNCODESTILLACTIVECONN **Meaning:** Environmental error. The requested structure cannot be deleted because it still has active or failed-persistent connections. **Action:** Do not delete a structure with active or failed-persistent connections. Use IXCQUERY to determine the state of the structure connections.

Table 45. Return and Reason Codes for the IXLFORCE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C29	**Equate Symbol:** IXLRSNCODEXESNOTACTIVE **Meaning:** Environmental error. The CFRM function is not active or is not available. **Action:** Bring the CFRM couple data set in use in the sysplex by issuing the SETXCF COUPLE,TYPE=CFRM,PCOUPLE= command.
C	xxxx0C2A	**Equate Symbol:** IXLRSNCODENOSUCHCONNECTION **Meaning:** Environmental error. The requested connection does not exist, or in the case of an ALLCONNS request, there are no failed-persistent connections to be deleted. **Action:** Ensure that the connection name was specified correctly.
C	xxxx0C33	**Equate Symbol:** IXLRSNCODECONNPENDINGRECONCIL **Meaning:** Environmental error. The specified structure still has connections in the coupling facility that are pending reconciliation into the CFRM active policy. **Action:** Retry the request at a later time.
C	xxxx0C4D	**Equate Symbol:** IXLRSNCODENOSTRDUMP **Meaning:** Environmental error. Either no structure dump was associated with the requested structure, or no structure dump with a matching STRDUMPID was associated with the requested structure. **Action:** Verify that the structure name and/or the structure dump identifier were specified correctly.
C	xxxx0C51	**Equate Symbol:** IXLRSNCODEREBUILDINPROGRESS **Meaning:** Environmental error. The requested structure is in a structure rebuild or a structure duplexing rebuild process. Structures and connections cannot be deleted while a structure rebuild is in progress. Structures and connections can be deleted while a structure duplexing rebuild is in the Duplex Established phase and no stop to switch to the new structure has been requested. **Action:** Retry the request at a later time.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular management protocol.
10	xxxx10xx	**Meaning:** Failure in XES processing. **Action:** Save the reason code information and contact the IBM support center.

IXLLIST — List Services

Introduction

A list structure is a coupling facility structure that enables multisystem application to share information organized as a set of lists and an optional set of locks. Each list consists of a queue of list entries.

The IXLLIST macro provides the means for applications in a sysplex to access and manipulate the data in a coupling facility list structure.

The IXLLIST macro provides many services. The service you require is designated by the REQUEST parameter. These services are detailed in the following chapters.

Starting with OS/390 Release 9, functions provided by the IXLLIST macro are supplemented with three additional macros — IXLLSTC, IXLLSTE, and IXLLSTM, that applications can use to access and manipulate the data in a coupling facility list structure. IXLLIST invocations will continue to be supported after OS/390 Release 8, but any new functionality will be added to the three new macros.

Be sure to read the IXLLIST guidance information in *z/OS MVS Programming: Sysplex Services Guide*. The information presented here assumes you have done so. The following table tells you where to find the reference information for each IXLLIST request type and indicates the OS/390 Release 9 new list services macro that provides the same, and possibly additional, functionality:

Table 46. Request Types for IXLLIST

Request Type	**Description**	**Page**
DELETE	Delete a list entry from a coupling facility list structure. You can also read an entry into your buffer and delete it from the coupling facility list structure. See also IXLLSTE REQUEST=DELETE.	671
DELETE_ENTRYLIST	Delete multiple list entries that are designated in an entry list contained in the storage area specified by BUFLIST or BUFFER. See also IXLLSTM REQUEST=DELETE_ENTRYLIST.	697
DELETE_MULT	Delete multiple list entries from a coupling facility list structure. See also IXLLSTM REQUEST=DELETE_MULT.	719
DEQ_EVENTQ	Read and dequeue event monitor controls from your event queue. See also IXLLSTC REQUEST=DEQ_EVENTQ.	735
LOCK	Perform locking functions such as SET, RESET, TEST, READNEXT. See also IXLLSTC REQUEST=LOCK.	751
MONITOR_EVENTQ	Start or stop monitoring your event queue for the presence of event monitor controls. See also IXLLSTC REQUEST=MONITOR_EVENTQ.	763
MONITOR_LIST	Start or stop monitoring a specified list for the presence of entries. See also IXLLSTC REQUEST=MONITOR_LIST.	775
MONITOR_SUBLIST	Start or stop monitoring a specified sublist for the presence of entries. See also IXLLSTC REQUEST=MONITOR_SUBLIST.	787
MONITOR_SUBLISTS	Start monitoring a set of sublists for the presence of entries. See also IXLLSTC REQUEST=MONITOR_SUBLISTS.	799

Table 46. Request Types for IXLLIST (continued)

Request Type	Description	Page
MOVE	Allows you to move an existing list entry from one list to another list or within the same list. You can also create a new list entry and place it on a list and write data to it. It is also possible to move an entry and read it into your buffer. See also IXLLSTE REQUEST=MOVE.	817
READ	Read a list entry from the coupling facility list structure into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA. See also IXLLSTE REQUEST=READ.	851
READ_EMCONTROLS	Retrieve control information about a sublist you are monitoring. See also IXLLSTC REQUEST=READ_EMCONTROLS.	877
READ_EQCONTROLS	Retrieve control information about your event queue. See also IXLLSTC REQUEST=READ_EQCONTROLS.	887
READ_LCONTROLS	Retrieve list control and monitoring information for a specified list. See also IXLLSTC REQUEST=READ_LCONTROLS.	897
READ_LIST	Read multiple entries from a single list into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA and ANSAREA. See also IXLLSTM REQUEST=READ_LIST.	911
READ_MULT	Read multiple entries from the coupling facility list structure into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA and ANSAREA. See also IXLLSTM REQUEST=READ_MULT.	935
WRITE	Write data from the storage areas specified by BUFFER or BUFLIST and/or ADJAREA to a list entry in the coupling facility list structure. See also IXLLSTE REQUEST=WRITE.	957
WRITE_LCONTROLS	Modify list controls for the specified list. This includes the list authority, list limit, list description, list cursor, list key, and maximum list key. See also IXLLSTC REQUEST=WRITE_LCONTROLS.	987

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued. This address space is referred to as the connector's address space. The requestor's address space refers to the home address space of the unit of work that is issuing the IXLLIST request. The requestor's address space may or may not be the same as the connector's address space.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Disabled callers must be legally disabled by holding the CPU lock and cannot hold other disabled locks. Enabled callers must not hold any locks. If MODE=SYNCSUSPEND is specified, the caller must be enabled.

Control parameters: See “Restrictions”

Programming Requirements

- If your program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLLIST. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.
- Include the IXLYCON mapping macro in your program. This macro provides a list of equate symbols for users of XES services and exits.
- Include mapping macros IXLYEMC, IXLYLAA, IXLYLCTL, IXLYLMI, and IXLYMSRI in your program as necessary. Table 47 lists these macros, the information and areas they map, and the particular IXLLIST requests and parameters they apply to.

Table 47. Mapping Macros for IXLLIST

Mapping Macro	Information	Area	Request/Parameter
IXLYEMC	Event monitor controls	BUFLIST buffers, BUFFER area	DEQ_EVENTQ
IXLYLAA	Answer area output	ANSAREA area	All requests
IXLYLCTL	List entry controls	Field LAALCTL of IXLYLAA	READ, WRITE, MOVE, DELETE, READ_LIST, READ_MULT
		Field LAARLRMLCTLS of IXLYLAA, BUFLIST buffers, BUFFER area	READ_LIST & READ_MULT (when TYPE=ECONTROLS)
IXLYLMI	List monitoring information	BUFLIST buffers, BUFFER area	READ_LCONTROLS
IXLYMSRI	Monitor sublists registration information	BUFLIST buffers, BUFFER area	MONITOR_SUBLISTS

Restrictions

- If you specify BUFLIST and PAGEABLE=YES, all of the buffers in the list must be in the same area of storage; you cannot mix common and private storage addresses for the buffers in the list.
- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).
- The caller’s parameter list must be addressable in the caller’s primary address space.
- If the caller is running in AR ASC mode and specifies a parameter using register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage areas specified by the ADJAREA, ANSAREA, and MOSVECTOR parameters must be addressable in the caller’s primary address space or from the caller’s PASN access list.
- The virtual storage areas specified by parameters other than ADJAREA, ANSAREA, and MOSVECTOR can be addressable in the caller’s primary, secondary, or home address spaces, from the PASN access list, or from the dispatchable unit access list (DU-AL).
- If the caller is disabled then the parameter list and all storage areas addressed by the parameters must reside in either nonpageable or disabled reference storage.

Input Register Information

Before issuing the IXLLIST macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code
1	Used as work register
2-13	Unchanged
14	Used as work register
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work register
2-13	Unchanged
14-15	Used as work register

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLLIST Version Support

The IXLLIST macro supports multiple versions — the version number corresponds to the level of the IXLLIST macro in which a keyword or function was introduced. The higher the version number, the more recent the version of the IXLLIST macro. Some IXLLIST keywords are supported for multiple versions, but have different functions for each version.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLLIST macro.
- The following IXLLIST keywords were available in the version 0 level of the macro, but have expanded function in the version 1 level.

AUTHCOMP	NEWAUTH

- The following IXLLIST keywords are supported by all versions starting with version 1 and higher of the IXLLIST macro:

KEYCOMP	LISTKEYINC	SETCURSOR
LISTKEY	MAXLISTKEY	

- The following keywords and functions are supported by all versions starting with version 2 and higher of the IXLLIST macro:

ENDINDEX	REQUEST=MONITOR_SUBLIST
MOSVECTOR	REQUEST=MONITOR_SUBLISTS
STARTINDEX	REQUEST=MONITOR_EVENTQ
UNC	REQUEST=READ_EMCONTROLS
	REQUEST=READ_EQCONTROLS
	REQUEST=DEQ_EVENTQ

- The following IXLLIST keyword is supported by all versions starting with version 3 and higher of the IXLLIST macro:

EXTRESTOKEN

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Coding Equivalent Functions

The following table displays examples of using version 1 keywords to achieve the equivalent function that was provided in version 0.

Table 48. IXLLIST Version Support

Version 1 Keyword	**Equivalent Function to Version 0**
AUTHCOMPTYPE	REQUEST=WRITE_LCONTROLS, AUTHCOMP=authcomp,AUTHCOMPTYPE=EQUAL
CURSORUPDTYPE	REQUEST=..., UPDATECURSOR=YES,CURSORUPDTYPE=NEXT
LISTKEYTYPE	REQUEST=WRITE/MOVE, LISTKEYTYPE=NOLISTKEY
VERSCOMPTYPE	REQUEST=..., VERSCOMP=verscomp,VERSCOMPTYPE=EQUAL

IXLLIST REQUEST=DELETE

Description

A DELETE request allows you to delete a single list entry from the list structure. Once removed from the list, the storage resources associated with the entry can be reused.

You can also specify that the designated entry be deleted only if it contains data that satisfies a version number comparison.

The entry you designate must exist on the list you specify, or the request fails with no change to the structure.

With a coupling facility of CFLEVEL=1 or higher, the following additional functions are supported for a DELETE request:

- You can specify that a list entry is to be deleted only after a successful comparison of the list authority value for the target list with a user specified value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. If the request is successful, you also can update the list authority value to a new value that you specify with NEWAUTH.
- Version number comparison is enhanced to allow for an additional equal-or-less-than-equal comparison operation.
- There are additional list cursor options. You can update the list cursor conditionally, and you can set the list cursor to point to the current entry instead of the previous or next entry.

On the DELETE request, you can optionally read data from the entry by specifying DATAOPER=READ. Before it deletes the entry, the system reads any combination of the following types of data for the designated entry:

- List entry controls
- Entry data
- Adjunct data

Whether a particular data type is returned also depends on whether the local storage area to contain that data is specified. Listed below is the data that is returned and the storage area that should be specified to contain that returned data:

- List entry controls, the number of list entries or elements remaining on the list, and the total number of allocated entries in the structure are returned to the answer area (ANSAREA).
- Entry data is returned to the buffer (BUFFER) or buffer list (BUFLIST). (DATAOPER=READ must be specified.)
- Adjunct data is returned to the adjunct area (ADJAREA). (DATAOPER=READ must be specified.)

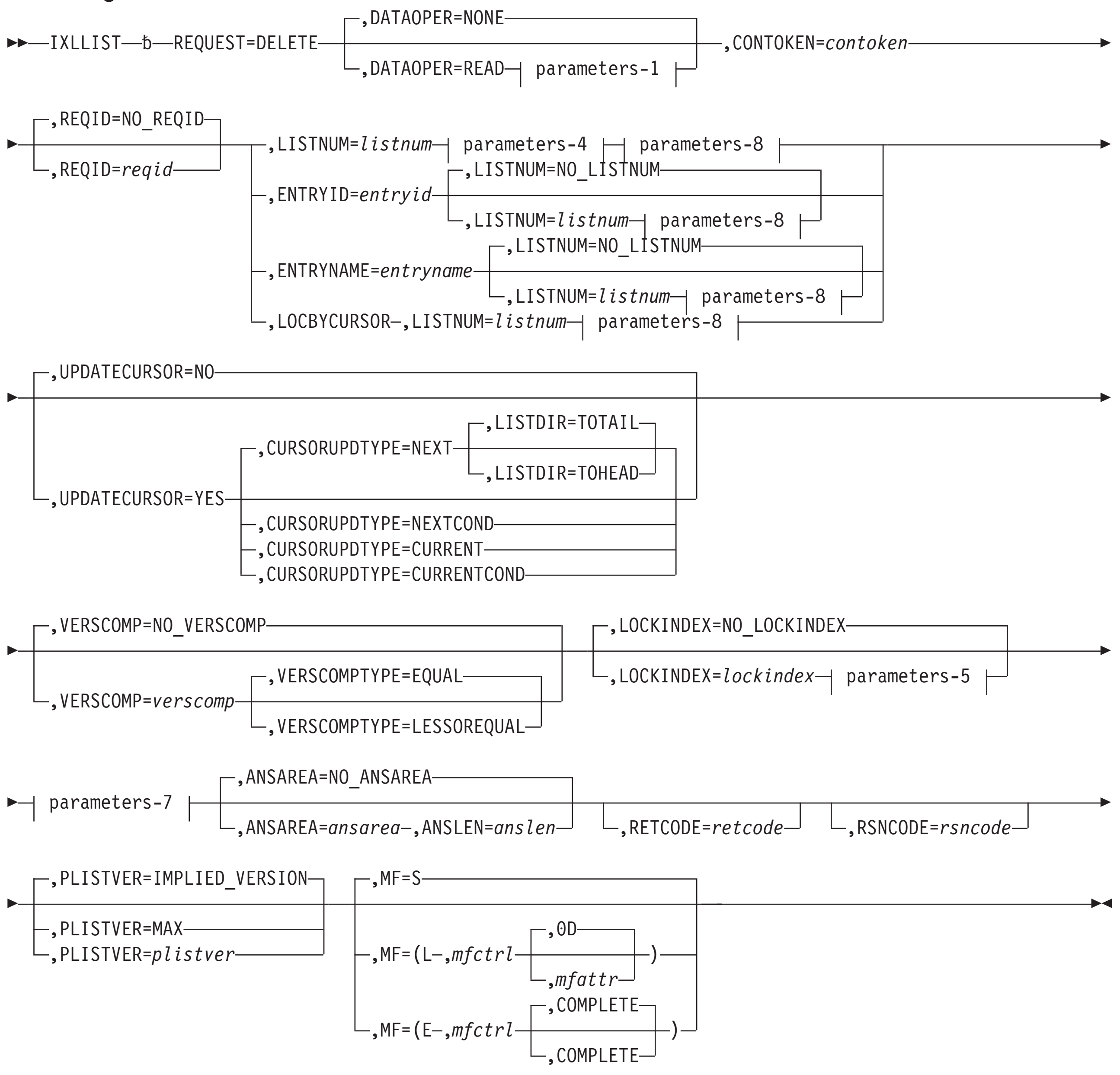
Additionally, you can perform locking functions with a DELETE request by specifying LOCKINDEX. The lock entry designated by LOCKINDEX will be operated on as specified by LOCKOPER.

Syntax Diagram

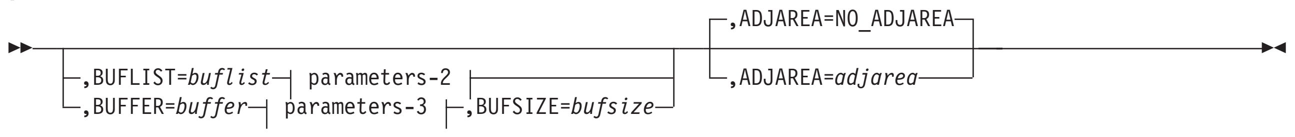
The syntax diagram for IXLLIST REQUEST=DELETE is as follows:

IXLLIST Macro

main diagram



parameters-1



parameters-2

```
     ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─┐
►►───┼──────────────────────────────────────────────────────────────────────────────┼──,BUFNUM=bufnum──────►
     │                                              ┌─,BUFALET=NO_BUFALET─┐         │
     ├─,BUFADDRTYPE=VIRTUAL──┤ parameters-3 ├──────┼─────────────────────┼─────────┤
     │                                              └─,BUFALET=bufalet────┘         │
     └─,BUFADDRTYPE=REAL────────────────────────────────────────────────────────────┘

►─,BUFINCRNUM=bufincrnum──────────────────────────────────────────────────────────────────────────────────►◄
```

parameters-3

```
     ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY───────┐
►►───┼───────────────────────────────────────────┼────────────────────────────────────────────────►◄
     │                  ┌─,BUFSTGKEY=CALLERS_KEY─┐│
     ├─,PAGEABLE=YES────┼────────────────────────┼┤
     │                  └─,BUFSTGKEY=bufstgkey───┘│
     └─,PAGEABLE=NO───────────────────────────────┘
```

parameters-4

```
     ┌─,LISTPOS=HEAD─┐  ┌─,ENTRYKEY=NO_ENTRYKEY───────────────────────────────────┐
►►───┼───────────────┼──┼─────────────────────────────────────────────────────────┼─────────────────►◄
     └─,LISTPOS=TAIL─┘  │                      ┌─,KEYREQTYPE=EQUAL──────────────┐ │
                        └─,ENTRYKEY=entrykey───┼────────────────────────────────┼─┘
                                               ├─,KEYREQTYPE=LESSOREQUAL────────┤
                                               └─,KEYREQTYPE=GREATEROREQUAL─────┘
```

parameters-5

```
►►──,LOCKOPER=──┬─SET──┤ parameters-6 ├───────────────┬─────────────────────────────────────────►◄
                │       ┌─,LOCKCOMP=NO_LOCKCOMP─┐      │
                ├─RESET─┼───────────────────────┼──────┤
                │       └─,LOCKCOMP=lockcomp────┘      │
                │         ┌─,LOCKMODE=UNCOND─┐         │
                ├─NOTHELD─┼──────────────────┼─────────┤
                │         └─,LOCKMODE=COND───┘         │
                │        ┌─,LOCKCOMP=NO_LOCKCOMP─┐     │
                └─HELDBY─┼───────────────────────┼─────┘
                         └─,LOCKCOMP=lockcomp────┘
```

parameters-6

```
     ┌─,LOCKMODE=UNCOND──┐   ┌─,LOCKDATA=NO_LOCKDATA─┐
►►───┼───────────────────┼───┼───────────────────────┼──────────────────────────────────────────►◄
     ├─,LOCKMODE=COND────┤   └─,LOCKDATA=lockdata────┘
     └─,LOCKCOMP=lockcomp┘
```

parameters-7

```
   ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►►─┼────────────────────────────────────────────┼─────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
   │                ┌─,REQDATA=NO_REQDATA─┐     │
   ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
   │                └─,REQDATA=reqdata────┘     │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
   │                 ┌─,REQDATA=NO_REQDATA─┐    │
   ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
   │                 └─,REQDATA=reqdata────┘    │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┤
   └─,MODE=ASYNCNORESPONSE──────────────────────┘
```

parameters-8

```
   ┌─,AUTHCOMP=NO_AUTHCOMP─────────────────────────────────┐  ┌─,NEWAUTH=NO_NEWAUTH─┐
►►─┼───────────────────────────────────────────────────────┼──┼─────────────────────┼──►◄
   │                       ┌─,AUTHCOMPTYPE=EQUAL───────┐   │  └─,NEWAUTH=newauth────┘
   └─,AUTHCOMP=authcomp────┼───────────────────────────┼───┘
                           └─,AUTHCOMPTYPE=LESSOREQUAL─┘
```

Notes:

1. In the | parameters-1 | fragment, one of the following must be specified:
   - BUFLIST=*buflist*
   - BUFFER=*buffer*
   - ADJAREA=*adjarea*

   In addition, ADJAREA=*adjarea* can be specified with either BUFLIST=*buflist* or BUFFER=*buffer*.
2. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.
3. If MODE=ASYNCNORESPONSE is specified, BUFFER, BUFLIST, and LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=DELETE are listed in alphabetical order. Default values are underlined:

REQUEST=DELETE
Use this input parameter to delete a single list entry from the list structure.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*
Use this output parameter to specify a storage area to contain the adjunct data that was read from the designated entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area to contain the adjunct data.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion of the request, the following information is returned to the answer area:
- List entry controls of the designated entry (field LAALCTL).
- The number of list entries or elements remaining on the list (field LAALISTCNT).
- The total number of allocated entries in the structure (field LAATOTALCNT).
- The total number of allocated elements in the structure (field LAATOTALELECNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information about the completion of the request will be put.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the entry data that is read from the designated entry.

You can define the buffer size to be a total of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

  See the BUFSIZE parameter description for defining the size of the buffer.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a size of BUFSIZE) that will contain the entry data.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold the entry data that is read from the designated entry. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:
- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:
- Reside in the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM parameter descriptions for specifying the number and size of buffers.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller’s PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CURSORUPDTYPE=NEXT
,CURSORUPDTYPE=NEXTCOND
,CURSORUPDTYPE=CURRENT
,CURSORUPDTYPE=CURRENTCOND
Use this input parameter to specify how the list cursor is to be updated when UPDATECURSOR=YES is specified.

Note: The NEXTCOND, CURRENT, and CURRENTCOND values are valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

CURSORUPDTYPE=NEXT
Update the list cursor to point to the list entry before or after the target entry.

- For MOVE requests that specify DATAOPER=WRITE and ENTRYTYPE=ANY and that result in the creation of a new entry, the cursor for the list specified by MOVETOLIST is updated. The direction of the cursor update depends on the value specified for MOVETOPOS.

- For all other requests, the cursor for the source list is updated. The direction of the cursor update depends on the value specified for LISTPOS or LISTDIR.

CURSORUPDTYPE=NEXTCOND
Update the list cursor to point to the list entry before or after the target entry only if the cursor points to the target list entry, and that list entry is moved or deleted.

Set the list cursor direction with SETCURSOR on a WRITE_LCONTROLS request.

The list cursor will be reset to binary zeros if either:

- The entry is the last entry on the list and the list cursor direction is set to a head-to-tail direction.
- The entry is the first entry on the list and the list cursor is set to a tail-to-head direction.

CURSORUPDTYPE=CURRENT
Set the list cursor to point to the list entry processed for the request.

If the list entry is deleted or moved to another list, then the list cursor will be reset to binary zeros.

CURSORUPDTYPE=CURRENTCOND
Set the list cursor to point to the list entry processed only if the list cursor value currently is zero and the target list entry is not deleted or moved to another list.

If the list entry is deleted or moved to another list, then the list cursor will remain binary zeros.

,DATAOPER=NONE
,DATAOPER=READ
Use this input parameter to specify an operation to be performed on the designated entry in addition to deleting it.

NONE
No additional operation is performed.

READ
Read and delete the entry data and/or adjunct data:

- If you have specified BUFFER or BUFLIST, the entry data is read into whichever buffer area you have specified.
- If you have specified ADJAREA, the adjunct data is read into the adjunct area specified.

,ENTRYID=*entryid*
Use this input parameter to designate the entry identifier of the entry to be deleted.

Each entry identifier, which is assigned by list services when the entry is created, is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=NO_ENTRYKEY
,ENTRYKEY=*entrykey*
Use this input parameter with LISTNUM to designate the entry key of the entry to be deleted.

If there is a sublist of entries on the list all with the same key as the ENTRYKEY key, the LISTPOS parameter determines whether the entry at the HEAD or TAIL of the sublist is designated.

Specify ENTRYKEY only for structures that support keyed entries.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*
Use this input parameter to designate the entry name of the entry to be deleted.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL
Use this input parameter to specify how, if at all, the entry key of the entry to be read can differ from the entry key specified by ENTRYKEY. KEYREQTYPE applies only to locating an existing entry; it does not determine the key of a new entry.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

Notes:

1. When no entries on the list meet the requirements of the KEYREQTYPE, the request will be failed with a return code X'8' and reason code IXLRSNCODEOENTRY.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, the entry with an entry key value closest to the value specified will be selected. When multiple entries have the same entry key value, LISTPOS is used to resolve whether the first or last entry with the entry key value is selected.

,LISTDIR=TOTAIL
,LISTDIR=TOHEAD
Use this input parameter with UPDATECURSOR to specify the direction in which the list cursor is moved as a result of this request. See the UPDATECURSOR parameter for a full description of LISTDIR.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*
Use this input parameter to specify the number of the list on which the entry to be deleted resides. Use LISTNUM in the following ways:

- With LISTPOS and/or ENTRYKEY to identify the entry to be deleted
- With either ENTRYID or ENTRYNAME to verify that the designated entry resides on the LISTNUM list before proceeding with the request
- With LOCBYCURSOR to identify the entry to be deleted

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTPOS=HEAD
,LISTPOS=TAIL
Use this input parameter with LISTNUM to designate the entry to be deleted. The designated entry is at the HEAD or the TAIL of a list or sublist:

- If you specify ENTRYKEY and the list contains a sublist of one or more entries with the same key (or that satisfies the KEYREQTYPE criteria), then:
  - LISTPOS=HEAD designates the entry at the head of the sublist.
  - LISTPOS=TAIL designates the entry at the tail of the sublist.
- If you do not specify ENTRYKEY, then:
  - LISTPOS=HEAD designates the entry at the head of the list.
  - LISTPOS=TAIL designates the entry at the tail of the list.

,LOCBYCURSOR

Use this input parameter to designate the list entry to be deleted. The designated entry is the entry to which the LISTNUM list cursor is pointing.

Be aware that the list cursor could have been reset to zeros by a previous request that, for instance, deleted or moved the entry to which the list cursor points, or updated the list cursor after the last entry on the list had been processed. See *z/OS MVS Programming: Sysplex Services Guide* for more information about using the list cursor.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA
,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:

- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND

Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND

If the specified lock is held by another user, your request is either:

- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND
The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY
Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET
Set the lock.

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released or fails, depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

RESET
Reset the lock.

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

NOTHELD
The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released or your request fails, depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

HELDBY
Processing is determined by which connector is holding the lock.

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*
Use this input parameter to specify a list authority value for the list specified by LISTNUM. If NEWAUTH is omitted, the list authority for the designated list is unchanged.

When the structure is allocated, the authority value for each list is initialized to binary zeros.

Note: The NEWAUTH parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher, except on WRITE_LCONTROLS requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,UPDATECURSOR=NO
,UPDATECURSOR=YES

Use this input parameter to specify whether the list cursor of the list containing the designated entry is to be updated to point to another entry on the list.

NO

The list cursor is not updated.

YES

The list cursor is updated to point to the entry that is adjacent to the entry that was deleted.

- If LISTDIR=TOHEAD, the list cursor will point to the adjacent entry that is closest to the head of the list.
- If LISTDIR=TOTAIL, the list cursor will point to the adjacent entry that is closest to the tail of the list.

The list cursor is set to binary zeros if its new position would be before the first entry or after the last.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of the designated entry to be deleted. The entry is deleted only if its version number meets the condition specified by the VERSCOMPTYPE parameter. If the designated entry does not have the specified version number, the request is terminated with no change to the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL

The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL

The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved, and the designated entry was deleted. The following fields in the answer area have been filled in: • LAATOTALCNT - The total count of allocated entries in the list structure. • LAATOTALELECNT - The total count of allocated elements in the list structure. • LAALISTCNT - The count of entries or elements residing on the processed list. • LAALCTL (if the request completed prematurely as well) - The list entry controls for the first UNPROCESSED entry in the list sequence. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=SET or LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller’s PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The designated entry has a version number that failed the comparison specified by VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the value specified in VERSCOMP and look at the version returned in the answer area.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by ENTRYID or ENTRYNAME exists in the structure, but does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND, or LOCKOPER=NOTHELD with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data for the first entry to be read in the list. No data is returned. **Action:** If more space is available, specify a larger buffer size, or more buffers, and reissue the request. Check the information returned in the answer area (mapped by IXLYLAA) in the field LAALCTL (mapped by IXLYLCTL).
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 49. Return and Reason Codes for IXLLIST REQUEST=DELETE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=DELETE_ENTRYLIST

Description

A DELETE_ENTRYLIST request allows you to delete multiple entries that are designated in an entry list contained in the storage area specified by BUFLIST or BUFFER. To designate the entries to be deleted, put either their entry identifier or entry name into the buffer (BUFFER) or buffer list buffers(BUFLIST). The LISTTYPE parameter indicates how the entries to be deleted are listed in the buffer. They may be listed by either entry identifier or by entry name, but not a combination of the two.

You can also request that a subset of the entries identified in the entry list be deleted by specifying the indexes into the entry list identifying the entries with which processing should begin (FIRSTELEM) and end (LASTELEM).

You can also restrict processing to only those entries that satisfy either one or both of the following criteria:

- Reside on a certain list (which you specify using LISTNUM)
- Contain data that satisfies a version number comparison (which you specify using VERSCOMP).

If you do not specify either LISTNUM or VERSCOMP, every entry between FIRSTELEM and LASTELEM will be deleted.

With a coupling facility of CFLEVEL=1 or higher, you can further restrict processing to only those entries that satisfy the following additional criteria:

- A successful comparison of the list authority value for the target list with a user-specified value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE.
- A successful comparison of the entry key value with a user-specified value (KEYCOMP). If the comparison is not successful, the current list entry is not deleted and processing continues with the next entry to be considered.
- An successful comparison of the version number, which is enhanced to allow for an additional equal-or-less-than-equal comparison operation.

When the DELETE_ENTRYLIST request completes, the number of deleted entries is returned in the answer area (ANSAREA).

When an entry identified in the entry list does not exist, or does not exist with the indicated version number and/or list number, processing is halted and the index of the failing entry is returned in the answer area. All the entries on the list preceding this entry are deleted; all succeeding entries have not yet been deleted. To continue processing, update FIRSTELEM and reissue the macro.

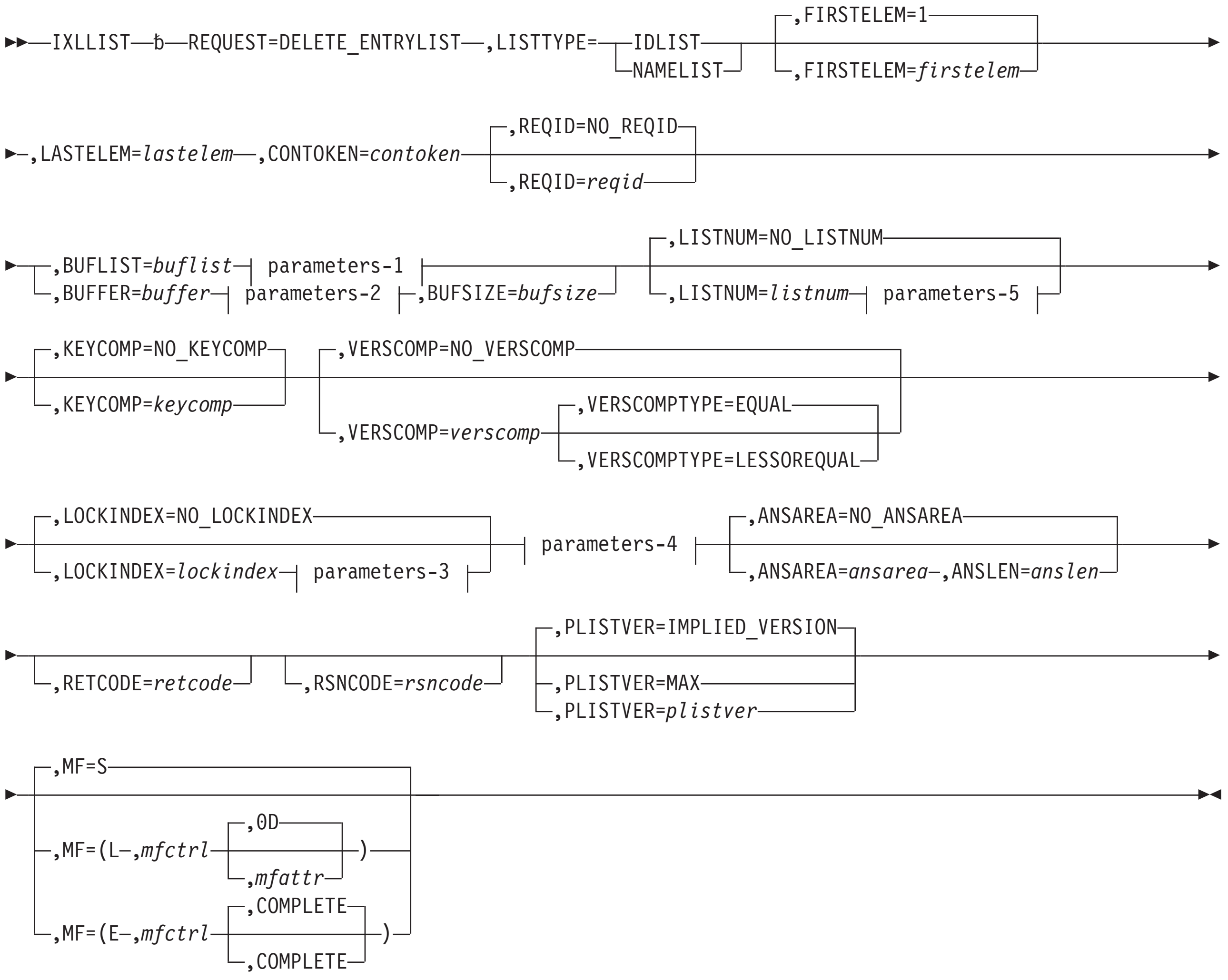
If the request completes prematurely because it exceeds the model-dependent time-out criteria, the number of deleted entries thus far and the index of the next entry to be deleted is returned in the answer area. You can update FIRSTELEM to this returned index (LAAFAILINDEX) on a subsequent DELETE_ENTRYLIST request to finish deleting the entries.

Syntax Diagram

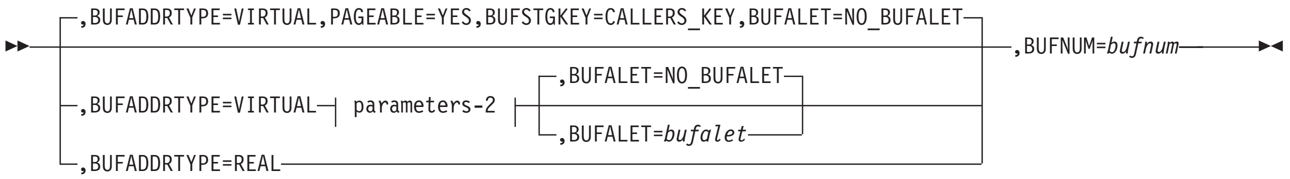
The syntax diagram for IXLLIST REQUEST=DELETE_ENTRYLIST is as follows:

IXLLIST Macro

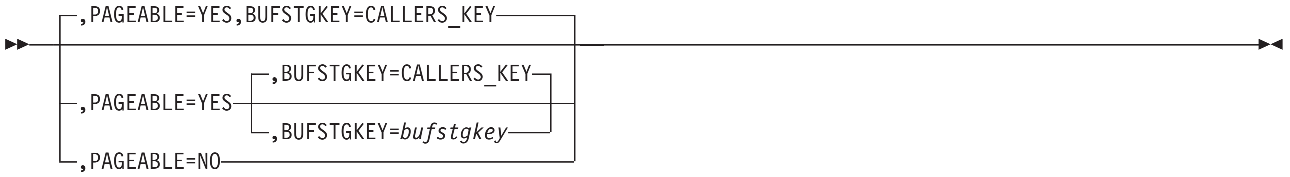
main diagram



parameters-1



parameters-2



parameters-3

```
►►─,LOCKOPER=─┬─NOTHELD─,LOCKMODE=COND──────────────┬──────────────────────────────►◄
              │          ┌─,LOCKCOMP=NO_LOCKCOMP─┐   │
              └─HELDBY──┼────────────────────────┼───┘
                         └─,LOCKCOMP=lockcomp─────┘
```

parameters-4

```
    ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►►──┼────────────────────────────────────────────┼─────────────────────────────────►◄
    ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
    │                ┌─,REQDATA=NO_REQDATA─┐     │
    ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
    │                └─,REQDATA=reqdata────┘     │
    ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
    ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
    │                 ┌─,REQDATA=NO_REQDATA─┐    │
    ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
    │                 └─,REQDATA=reqdata────┘    │
    ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┤
    └─,MODE=ASYNCNORESPONSE──────────────────────┘
```

parameters-5

```
    ┌─,AUTHCOMP=NO_AUTHCOMP──────────────────────────────┐
►►──┼────────────────────────────────────────────────────┼─────────────────────────►◄
    │                      ┌─,AUTHCOMPTYPE=EQUAL───────┐ │
    └─,AUTHCOMP=authcomp───┼───────────────────────────┼─┘
                           └─,AUTHCOMPTYPE=LESSOREQUAL─┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, BUFFER, BUFLIST, and LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=DELETE_ENTRYLIST are listed in alphabetical order. Default values are underlined:

REQUEST=DELETE_ENTRYLIST
Use this input parameter to delete multiple entries that are identified in an entry list.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

When the request completes successfully, the answer area contains the number of deleted entries (field LAADELCNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information returned by the request.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*
Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to hold either the entry identifiers or the entry names of the list entries to be deleted. Whether you place entry identifiers or entry names in the buffer depends on the LISTTYPE value you specify:
- If you specify LISTTYPE=IDLIST, the buffer should contain list entry identifiers.
- If you specify LISTTYPE=NAMELIST, the buffer should contain list entry names.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes.
- Is less than or equal to 65536 bytes.
- Starts on a 4096-byte boundary.

The buffer should be formatted according to the LISTTYPE value you specify:
- If LISTTYPE=IDLIST, format the buffer into units of 12 bytes each. Each 12-byte unit should contain the entry identifier of one entry to be deleted.
- For LISTTYPE=NAMELIST, format the buffer into units of 16-bytes each. Each 16-byte unit should contain the entry name of one entry to be deleted.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry identifiers or entry names.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold the entry identifiers or the entry names of the list entries to be deleted. Whether you place entry identifiers or entry names in the buffers depends on the LISTTYPE value you specify:
- If you specify LISTTYPE=IDLIST, the buffers should contain list entry identifiers.
- If you specify LISTTYPE=NAMELIST, the buffers should contain list entry names.

BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:
- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:
- Reside in the same address space or data space.
- Be 4096 bytes in length.
- Start on a 4096-byte boundary.

  Note: The buffers do not have to be contiguous in storage. (List services treats BUFLIST buffers as a single, contiguous buffer, even if the buffers are not contiguous.)

  See the BUFNUM parameter description for specifying the number of buffers in the buffer list.

The buffers must be arranged according to the LISTTYPE value you specify:
- If LISTTYPE=IDLIST, each buffer should contain elements of 12 bytes each. Each 12-byte element should contain the entry identifier of one entry to be deleted.
- For LISTTYPE=NAMELIST, each buffer should contain elements of 16 bytes each. Each 16-byte element should contain the entry name of one entry to be deleted.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no entries will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,FIRSTELEM=1
,FIRSTELEM=*firstelem*
Use this input parameter to specify an index into the entry list. The FIRSTELEM index identifies where in the list of entries to begin processing. A FIRSTELEM index of 5, for example, indicates that the fifth entry in the list is the beginning of the list of entries to be deleted.

The value must identify one of the entry identifiers or entry names contained in the BUFFER area or BUFLIST buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the entry index.

,KEYCOMP=NO_KEYCOMP
,KEYCOMP=*keycomp*
Use this input parameter to specify a value to be compared to the entry key of each list entry in the designated list.

If the list entry that is to be processed does not have an entry key value that is equal to the specified KEYCOMP value, the entry is not processed. Processing continues with the next entry.

Note: The KEYCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the value to be compared to the entry key of the designated list entry.

,LASTELEM=*lastelem*

Use this input parameter to specify an index into the entry list. The LASTELEM index identifies the entry that designates the end of the list or the last entry that you want to process. A LASTELEM index of 10, for example, indicates that the entry associated with the tenth entry identifier or entry name in the entry list will be the final entry to be deleted.

The LASTELEM index must be greater than or equal to the FIRSTELEM index, and must specify one of the entry identifiers or entry names contained in the BUFFER area or BUFLIST buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the entry index.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entries to be deleted must reside. Specifying a list number restricts processing to entries that reside on the designated list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTTYPE=IDLIST
,LISTTYPE=NAMELIST

Use this input parameter to specify whether the input data in the BUFFER area or BUFLIST buffers consists of a list of entry identifiers or entry names.

IDLIST
The contents of the BUFFER area or BUFLIST buffers is a list of entry identifiers.

NAMELIST
The contents of the BUFFER area or BUFLIST buffers is a list of entry names.

Note: You cannot specify LISTTYPE=NAMELIST for structures that do not support named entries.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=COND

Use this input parameter to specify that the lock operation for the lock specified by LOCKINDEX be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

LOCKOPER=NOTHELD

The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released, or your request fails depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

LOCKOPER=HELDBY

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro.

For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*
Use this input parameter to specify a version number to be compared to the version number of each entry in the list of entries to be deleted. Only those entries with a version that meets the condition specified by the VERSCOMPTYPE parameter will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request completed prematurely because it exceeded the coupling facility model dependent time-out criteria. The following information has been returned in the answer area: • The number of entries that were deleted (field LAADELCNT) • The index of the first unprocessed entry identifier or entry name (field LAAFAILINDEX) **Action:** Reissue the request specifying for FIRSTELEM the index of the first unprocessed entry. For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=SET or LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The value specified for either FIRSTELEM or LASTELEM was not valid. Some entries may have been processed. The answer area (mapped by IXLYLAA) contains: • The count of entries successfully deleted, field LAADELCNT. • The index into the entry list of the entry identifier or entry or entry name that was not valid, field LAAFAILINDEX. **Action:** Ensure that the FIRSTELEM and LASTELEM values correspond to entries in the list of entries to be processed.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0843	**Equate Symbol:** IXLRSNCODEBADENTRYID **Meaning:** Program error. The entry corresponding to either an entry identifier (LISTTYPE=IDLIST)or entry name (LISTTYPE=NAMELIST) in BUFFER or the BUFLIST buffers does not exist. The index to the failing entry identifier or name was returned in the answer area (field LAAFAILINDEX). The count of entries deleted is also returned in the answer are (field LAADELCNT). **Action:** Remove the failing entry from the list, or reissue the request. Update FIRSTELEM to point after the failing entry (FIRSTELEM = LAAFAILINDEX + 1) to the next entry on the list to be processed.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.

Table 50. Return and Reason Codes for IXLLIST REQUEST=DELETE_ENTRYLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=DELETE_MULT

Description

A DELETE_MULT request allows you to delete multiple entries from the list structure. Once removed, the storage resource associated with the entries can be reused. You can restrict processing to only those entries that satisfy either one or both of the following criteria:

- Reside on a certain list (which you specify using LISTNUM).
- Contain data that satisfies a version number comparison (which you specify using VERSCOMP).

Important: For list structures allocated in a CFLEVEL=0 coupling facility, if you do not specify either LISTNUM or VERSCOMP, every entry in the structure is deleted.

With a coupling facility of CFLEVEL=1 or higher, you can restrict processing to only those entries that satisfy one or more of the following additional criteria:

- A successful comparison of the list authority value for the target list with a user-specified value. You specify the authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. You must also specify the list number using LISTNUM.
- A successful comparison of the entry key value with a user-specified value (KEYCOMP). If the comparison is not successful, the current list entry is not deleted and processing continues with the next entry to be considered.
- A version number comparison that is enhanced to allow for an additional equal-or-less-than-equal comparison operation.

If the request completes prematurely because it exceeds the coupling facility model-dependent time-out criteria, a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned to the answer area (ANSAREA). The token can be specified on another DELETE_MULT request to resume processing with the next entry to be deleted. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

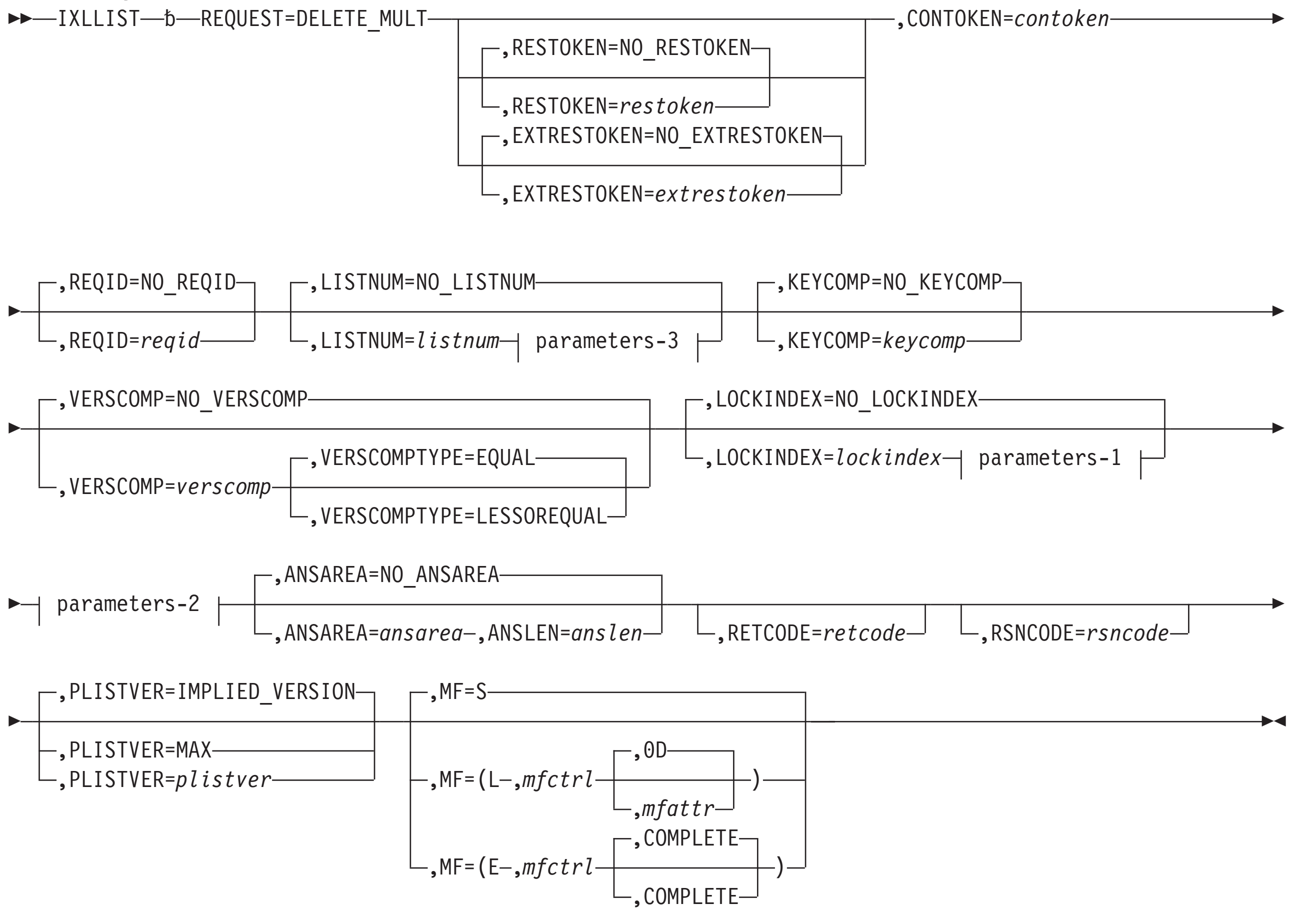
When the DELETE_MULT request completes, the number of entries that were deleted is returned to the answer area.

Syntax Diagram

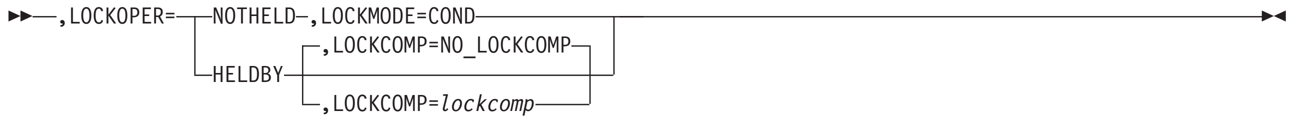
The syntax diagram for IXLLIST REQUEST=DELETE_MULT is as follows:

IXLLIST Macro

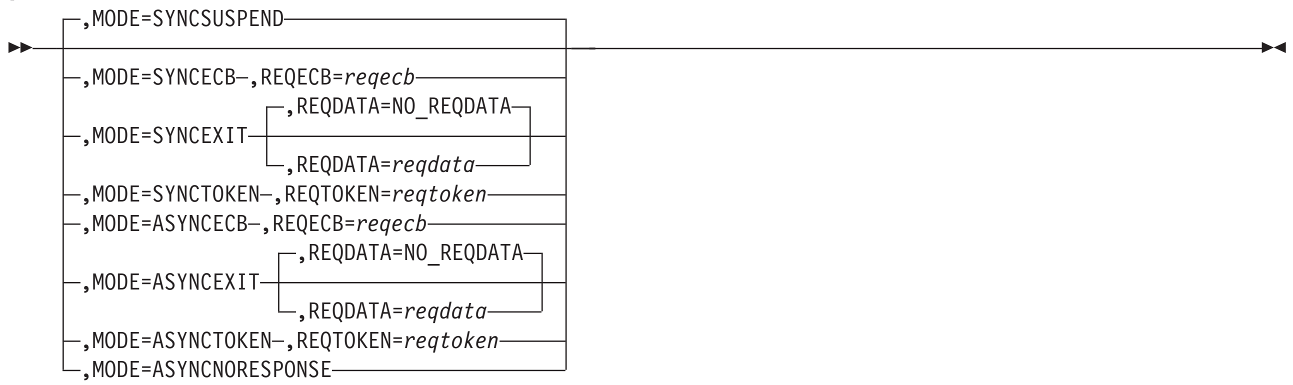
main diagram



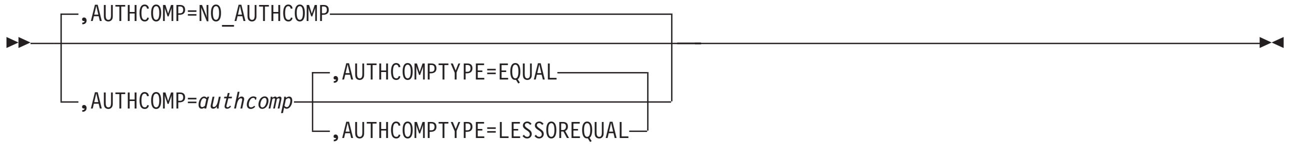
parameters-1



parameters-2



parameters-3



Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=DELETE_MULT are listed in alphabetical order. Default values are underlined:

REQUEST=DELETE_MULT
Use this input parameter to delete multiple entries from the list structure.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

When the request completes, the answer area contains the number of deleted entries (field LAADELCNT).

If the request completes prematurely, a restart token (field LAARESTOKEN) or extended restart token (field LAAEXTRESTOKEN) is returned that may be specified on a subsequent DELETE_MULT request. (See the RESTOKEN and EXTRESTOKEN parameter descriptions.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information related to the request.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*
Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*
Use this input parameter to specify an extended restart token that can be used to resume processing of a DELETE_MULT request that completed prematurely. The extended restart token is returned in the answer area (field LAAEXTRESTOKEN), and should be specified on the next DELETE_MULT request to resume processing.

If the request does not complete prematurely, the extended restart token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes list services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,KEYCOMP=NO_KEYCOMP
,KEYCOMP=*keycomp*
Use this input parameter to specify a value to be compared to the entry key of each list entry in the designated list.

If the list entry that is to be processed does not have an entry key value that is equal to the specified KEYCOMP value, the entry is not processed. Processing continues with the next entry.

Note: The KEYCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the value to be compared to the entry key of the designated list entry.

,LISTNUM=NO_LISTNUM

,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entries to be deleted must reside. Specifying a list number restricts delete processing to entries that reside on the designated list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=COND

Use this input parameter to specify that the lock operation for the lock specified by LOCKINDEX be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

LOCKOPER=NOTHELD

The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released, or your request fails depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

LOCKOPER=HELDBY

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**

,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*
Use this input parameter to specify a restart token that can be used to resume processing of a DELETE_MULT request that completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The restart token is returned in the answer area (field LAARESTOKEN), and should be specified on the next DELETE_MULT request to resume processing with the next entry to be deleted.

If the request does not exceed the model-dependent time-out criteria, the returned token will not be provided.

Notes:
1. Specifying an extended restart token of all zeros causes list services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VERSCOMP=NO_VERSCOMP

,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of each entry to be deleted. Only those entries with a version that meets the condition specified by the VERSCOMPTYPE parameter will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL

The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL

The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of entries that were deleted (field LAADELCNT) • A token for restarting the request (field LAARESTOKEN or LAAEXTRESTOKEN) **Action:** Reissue the request using the restart token (RESTOKEN or EXTRESTOKEN). For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to restart the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request (field LAARESTOKEN). • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. **Action:** Reinitiate the request with an EXTRESTOKEN value of zero. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. Discard the results of the initial request and reissue the request. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.

Table 51. Return and Reason Codes for IXLLIST REQUEST=DELETE_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=DEQ_EVENTQ

Description

A DEQ_EVENTQ request allows you to read and dequeue event monitor controls from your event queue. The EMCs are returned in the storage area specified by the BUFFER or BUFLIST parameter. The information for each EMC object is mapped by the IXLYEMC macro

A DEQ_EVENTQ request can complete before all the events have been read off your event queue. If this occurs, you are notified by a return code indicating that more events remain on the event queue. The count of EMCs that are still queued to the event queue is returned in the answer area.

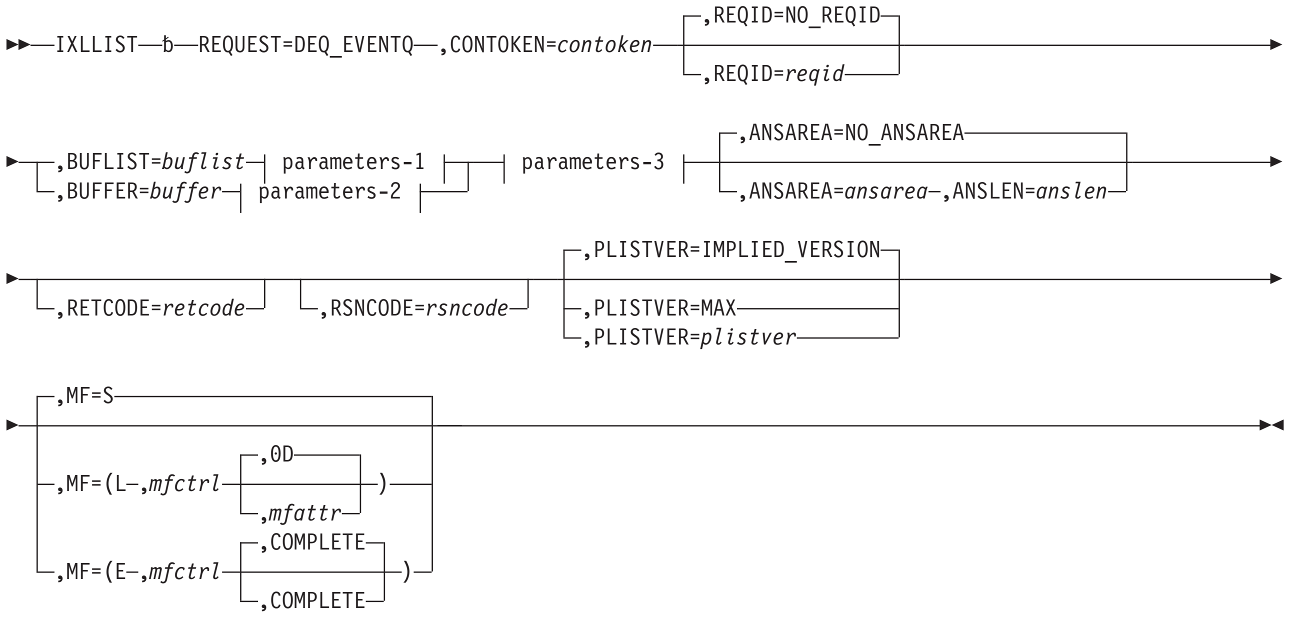
In the case of a premature completion of an IXLLIST REQUEST=DEQ_EVENTQ request, all prior EMCs were processed successfully. To continue to read and dequeue EMCs from your event queue, reissue the IXLLIST REQUEST=DEQ_EVENTQ as many times as is required after processing the previous contents of the BUFFER or BUFLIST storage area.

A DEQ_EVENTQ request can be issued only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. DEQ_EVENTQ requests issued for a list structure allocated in a coupling facility of CFLEVEL 0, 1, or 2 will fail.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=DEQ_EVENTQ is as follows:

main diagram



parameters-1

```
    ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─┐
►►──┼─────────────────────────────────────────────────────────────────────────┼──────────────────►◄
    │                                               ┌─,BUFALET=NO_BUFALET─┐    │
    ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├─────────┼─────────────────────┼────┤
    │                                               └─,BUFALET=bufalet────┘    │
    └─,BUFADDRTYPE=REAL────────────────────────────────────────────────────────┘
```

parameters-2

```
    ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────────┐
►►──┼──────────────────────────────────────────────┼──────────────────────────────────────────►◄
    │                ┌─,BUFSTGKEY=CALLERS_KEY─┐    │
    ├─,PAGEABLE=YES──┼────────────────────────┼────┤
    │                └─,BUFSTGKEY=bufstgkey───┘    │
    └─,PAGEABLE=NO─────────────────────────────────┘
```

parameters-3

```
    ┌─,MODE=SYNCSUSPEND──────────────────────────────┐
►►──┼────────────────────────────────────────────────┼────────────────────────────────────────►◄
    ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤
    │                 ┌─,REQDATA=NO_REQDATA─┐        │
    ├─,MODE=SYNCEXIT──┼─────────────────────┼────────┤
    │                 └─,REQDATA=reqdata────┘        │
    ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
    ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
    │                  ┌─,REQDATA=NO_REQDATA─┐       │
    ├─,MODE=ASYNCEXIT──┼─────────────────────┼───────┤
    │                  └─,REQDATA=reqdata────┘       │
    ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┤
    └─,MODE=ASYNCNORESPONSE──────────────────────────┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=DEQ_EVENTQ are listed in alphabetical order. Default values are underlined:

REQUEST=DEQ_EVENTQ

Use this input parameter to read and dequeue event monitor controls (EMCs) from the user's event queue.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

When the request completes successfully or completes prematurely, the answer area contains the number of event monitor controls still queued to the event queue (field LAADEQ_EMCQUEUEDCNT) and the number of event monitor controls that were read and dequeued (field LAADEQ_NUMEMCREAD).

See *z/OS MVS Programming: Sysplex Services Guide* for a description of all the relevant IXLYLAA fields for this request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information returned by the request.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL

The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL

The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the event monitor controls (EMC) objects that have been read and dequeued from your event queue.

You must define the buffer to be 4096 bytes on a 4096-byte boundary. The buffer must not start below storage address 512.

Once the request completes, the buffer contains, starting at offset zero, an array of event monitor controls that were dequeued from the user’s event queue. The format of each array element is described by mapping macro IXLYEMC.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4096-byte area to contain the data returned by the request.

,BUFLIST=*buflist*

Use this input parameter to specify a buffer to hold the returned event monitor controls (EMC) objects that have been read and dequeued from your event queue.

BUFLIST specifies a 128-byte storage area that contains the address of a single buffer. The address of the buffer should be placed in the second four bytes of the 128-byte area, beginning at offset zero. The first four bytes are reserved and the rest of the area is ignored.

You must define the buffer to be 4096 bytes on a 4096-byte boundary. The buffer must not start below storage address 512.

Once the request completes, the buffer contains, starting at offset zero in the buffer, an array of event monitor controls that were dequeued from the user's event queue. The format of each array element is described by the mapping macro IXLYEMC.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the address of the buffer.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN

The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB

The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT

The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN

The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE

The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR

C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of event monitor controls that were read and dequeued from the event queue (field LAADEQ_NUMEMCREAD). • The number of event monitor controls that remain on the event queue (field LAADEQ_EMCQUEUEDCNT). **Action:** After processing all returned data, reissue the request to continue. For more information about premature completion, of a DEQ_EVENTQ request, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size, and reissue the request.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • The requirements for BUFLIST or BUFFER. • Buffer pointer(s) in the BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.

Table 52. Return and Reason Codes for IXLLIST REQUEST=DEQ_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=LOCK

Description

A LOCK request allows you to perform locking functions. The lock designated by LOCKINDEX is operated on as specified by the LOCKOPER parameter. You can use REQUEST=LOCK only for structures that contain a lock table (the LOCKENTRIES parameter on the IXLCONN macro causes a list structure to have a lock table.)

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=LOCK is as follows:

main diagram

```
                                                  ┌─,REQID=NO_REQID─┐
►►──IXLLIST──␢──REQUEST=LOCK──,CONTOKEN=contoken──┼─────────────────┼──,LOCKINDEX=lockindex──────────►
                                                  └─,REQID=reqid────┘

                                                   ┌─,MODE=SYNCSUSPEND─────────────────────────┐
►─,LOCKOPER=──┬─SET─┤ parameters-1 ├──────────────┬┼───────────────────────────────────────────┼──────►
              │      ┌─,LOCKCOMP=NO_LOCKCOMP─┐     │├─,MODE=SYNCECB─,REQECB=reqecb──────────────┤
              ├─RESET┼───────────────────────┼─────┤│              ┌─,REQDATA=NO_REQDATA─┐      │
              │      └─,LOCKCOMP=lockcomp────┘     │├─,MODE=SYNCEXIT┼────────────────────┼──────┤
              │     ┌─,LOCKCOMP=NO_LOCKCOMP─┐      ││              └─,REQDATA=reqdata────┘      │
              ├─TEST┼───────────────────────┼──────┤├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken──────────┤
              │     └─,LOCKCOMP=lockcomp────┘      │├─,MODE=ASYNCECB─,REQECB=reqecb─────────────┤
              │         ┌─,LOCKCOMP=NO_LOCKCOMP─┐  ││                ┌─,REQDATA=NO_REQDATA─┐    │
              └─READNEXT┼───────────────────────┼──┘├─,MODE=ASYNCEXIT┼────────────────────┼────┤
                        └─,LOCKCOMP=lockcomp────┘   │                └─,REQDATA=reqdata────┘    │
                                                    └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken─────────┘

   ┌─,ANSAREA=NO_ANSAREA───────────────┐
►──┼───────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬──────────────────►
   └─,ANSAREA=ansarea─,ANSLEN=anslen───┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S────────────────────────────────┐
►──┼───────────────────────────┼──┼──────────────────────────────────────┼────────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D──────┐         │
   └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl───┼──────────┼─)──────┤
                                  │                  └─,mfattr──┘         │
                                  │                  ┌─,COMPLETE─┐        │
                                  └─,MF=(E─,mfctrl───┼───────────┼─)──────┘
                                                     └─,COMPLETE─┘
```

parameters-1

```
    ┌─,LOCKMODE=UNCOND────┐  ┌─,LOCKDATA=NO_LOCKDATA─┐
►►──┼─────────────────────┼──┼───────────────────────┼────────────────────────────────────────────►◄
    ├─,LOCKMODE=COND──────┤  └─,LOCKDATA=lockdata────┘
    └─,LOCKCOMP=lockcomp──┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=LOCK are listed in alphabetical order. Default values are underlined:

REQUEST=LOCK

Use this input parameter to specify that the LOCKINDEX lock be operated on as specified by the LOCKOPER parameter.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request for LOCKOPER=READNEXT request, the answer area contains the connection ID of the connection holding the lock (field LAACONID) and the index of the lock found for a request (field LAALOCKINDEX). For a request with LOCKOPER=TEST, the answer area contains the connection ID of the connection holding the lock (field LAACONID). IXLYLAA fields for this request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA
,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:

- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKINDEX=*lockindex*
Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. Note that lock indexes begin with zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND
Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND
If the specified lock is held by another user, your request is either:

- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND
The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=TEST
,LOCKOPER=READNEXT
Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX, or for READNEXT, beginning with the lock specified by LOCKINDEX.

LOCKOPER=SET

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released, or fails depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

LOCKOPER=RESET

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

LOCKOPER=TEST

- When LOCKCOMP is not specified, the lock is tested to check if it is held by your connection.
- When LOCKCOMP is also specified, the lock is tested to check if it is held by the connection specified by LOCKCOMP.

If the system returns a return code of zero, the lock was held by the specified user. If the system returns a return code of X'4' and a reason code of IXLRSNCODELOCKNOTHELD, IXLRSNCODELOCKCOND, or IXLRSNCODELOCKHELDBYSYS, the lock is either not held or is held by a user other than the specified user.

LOCKOPER=READNEXT
This option scans the lock table, beginning with the lock index specified by LOCKINDEX, and returns to the answer area information as follows:

- When LOCKCOMP is not specified, the lock index of the next held lock, and the connection identifier of the owner of that lock, is returned.
- When LOCKCOMP is specified, the lock index of the next lock held by the LOCKCOMP connection is returned.

The request might end prematurely because it exceeded the coupling facility model-dependent time-out criteria. If this happens, the index of the next lock to be processed by the request is returned in the answer area (LAALOCKINDEX). This returned lock index can be specified for LOCKINDEX on another IXLLIST REQUEST=LOCK request to resume processing of the lock table. Except for LOCKINDEX, the same parameters and values should be specified on the resumed request as on the previous request.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT

,MODE=ASYNCTOKEN
Use this input parameter to specify whether the request is to be performed synchronously or asynchronously. MODE can also be used to specify how your program can be notified of request completion.

MODE=SYNCSUSPEND
The request is performed synchronously. If necessary, the caller is suspended. Control is returned to the caller when the request completes. The caller must be in an enabled state to use this option.

MODE=SYNCECB
The request attempts synchronous processing. If the request cannot be completed synchronously, control is returned to the caller before the request completes, and the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request attempts synchronous processing. If the request cannot be completed synchronously, control is returned to the caller before the request completes, and the connection's complete exit is called when the request completes.

MODE=SYNCTOKEN
The request attempts synchronous processing. If the request cannot be completed synchronously, control is returned to the caller before the request completes, and a token that uniquely identifies the request is returned in the area specified by REQTOKEN. The token is required to force completion of the request. See the REQTOKEN parameter description for a complete explanation of the request token.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
Control is returned to the caller before the request completes. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
Control is returned to the caller before the request completes. The connection's complete exit is called when the request completes.

MODE=ASYNCTOKEN
Control is returned to the caller before the request completes. A token that uniquely identifies the request is returned to the area specified by REQTOKEN. The token is required to force completion of the request. See the REQTOKEN parameter description for a complete explanation of the request token.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** • If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. • If LOCKOPER=TEST was specified, this return code indicates the lock is held by the connection specified by CONTOKEN. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed. • If you specified MODE=ASYNCNORESPONSE, no action is required. You will not be notified when the request completes.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request specifying LOCKOPER=READNEXT completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The index of the next lock to be processed has been returned in the answer area (field LAALOCKINDEX). **Action:** Reissue the request specifying for LOCKINDEX the index of the next lock to be processed. For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040E	**Equate Symbol:** IXLRSNCODELOCKNOTHELD **Meaning:** A LOCKOPER=TEST request determined that the specified lock was not held for the specified connection. The connection ID of the lock owner is returned in the answer area (field LAACONID). **Action:** None necessary. If this reason code is not expected, determine who holds the lock and see if any clean-up is necessary. The lock may need to be released, or you can wait and try again.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified LOCKMODE=COND or LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. For a request that specified LOCKOPER=READNEXT, the request found a lock that is not generally available (see field LAALOCKINDEX in the answer area). **Action:** Retry the request, or obtain the lock as required, and retry the request.
4	xxxx041F	**Equate Symbol:** IXLRSNCODENOLOCKSHELD **Meaning:** A request specifying LOCKOPER=READNEXT found no locks held from the LOCKINDEX lock to the end of the lock table. **Action:** None necessary.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Cod	Equate Symbol Meaning and Action
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Cod	Equate Symbol Meaning and Action
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 53. Return and Reason Codes for IXLLIST REQUEST=LOCK Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Cod**	**Equate Symbol Meaning and Action**
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=MONITOR_EVENTQ

Description

A MONITOR_EVENTQ request allows you to start (ACTION=START) or stop (ACTION=STOP) monitoring your event queue for the presence of event monitor controls (EMCs). The system queues or withdraws event monitor controls on your event queue when sublist monitoring is in effect. The EMCs indicate the empty or nonempty state of the sublist(s) that you are monitoring.

When starting event queue monitoring, you must specify the list notification vector index (VECTORINDEX) to be associated with the event queue. List services uses the specified vector index to indicate whether the event queue is in the **empty** or **non-empty** state.

If you want to be notified when the event queue for which monitoring has been activated has changed state from empty to non-empty, specify DRIVEEXIT=YES. This option causes your list transition exit to be driven, informing you that your monitored event queue now has at least one entry on it. You must identify your list transition exit on the LISTTRANEXIT parameter of the IXLCONN macro. You can use the IXLVECTR macro to test the vector entry that you have associated with your monitored event queue.

A MONITOR_EVENTQ request can be issued only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. MONITOR_EVENTQ requests issued for a list structure allocated in a coupling facility of CFLEVEL 0, 1, or 2 will fail.

If you have started event queue monitoring with a certain vector index specified and you want to monitor the event queue with a different vector index, do not stop monitoring the event queue with the old vector index. Simply start monitoring the event queue with the new index; the new index will automatically replace the old.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=MONITOR_EVENTQ is as follows:

main diagram

```
►►──IXLLIST──␢──REQUEST=MONITOR_EVENTQ─────────────────────────────────────────────────►

                                                 ┌─,DRIVEEXIT=YES─┐
►─,ACTION=─┬─START─,VECTORINDEX=vectorindex─┴─,DRIVEEXIT=NO──┴─┬─,CONTOKEN=contoken──────►
           └─STOP─────────────────────────────────────────────┘

  ┌─,REQID=NO_REQID─┐                    ┌─,ANSAREA=NO_ANSAREA──────────────────┐
►─┴─,REQID=reqid────┴─┤ parameters-1 ├─┴─,ANSAREA=ansarea─,ANSLEN=anslen─┴─┬───────────────────┬─►
                                                                             └─,RETCODE=retcode─┘

                         ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►─┬───────────────────┬─┼─,PLISTVER=MAX─────────────┼──┼────────────────────────────────────────┴──►◄
  └─,RSNCODE=rsncode─┘  └─,PLISTVER=plistver───────┘  │                  ┌─,0D─────┐            │
                                                      ├─,MF=(L─,mfctrl─┴─,mfattr─┴─)──────────┤
                                                      │                  ┌─,COMPLETE─┐          │
                                                      └─,MF=(E─,mfctrl─┴─,COMPLETE─┴─)────────┘
```

parameters-1

```
    ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►►──┼────────────────────────────────────────────┼─────────────────────────────────────────────────►◄
    ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
    │                ┌─,REQDATA=NO_REQDATA─┐     │
    ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
    │                └─,REQDATA=reqdata────┘     │
    ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
    ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
    │                 ┌─,REQDATA=NO_REQDATA─┐    │
    ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
    │                 └─,REQDATA=reqdata────┘    │
    ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┤
    └─,MODE=ASYNCNORESPONSE──────────────────────┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=MONITOR_EVENTQ are listed in alphabetical order. Default values are underlined:

REQUEST=MONITOR_EVENTQ
Use this input parameter to specify that monitoring of the event queue associated with this user be started or stopped.

,ACTION=START
,ACTION=STOP
Use this input parameter to specify whether event queue monitoring is to be started or stopped.

START
Start monitoring the event queue or restart monitoring with different parameters.

STOP
Stop monitoring the event queue.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request with ACTION=START, the answer area contains the number of event monitor controls (EMCs) queued to your event queue when monitoring was established (field LAAMNEQ_EVENTCNT) and the state (empty or nonempty) of the event queue (field LAAMNEQ_EVENTQUEUED).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DRIVEEXIT=YES
,DRIVEEXIT=NO

Use this input parameter to specify whether list services should drive the connection's list transition exit when the state of the event queue changes from empty to nonempty. The list transition exit informs the connection that the event queue being monitored has changed state. Use the IXLVECTR macro to test the vector entry that you have associated with your monitored event queue.

YES

The connection's list transition exit will be called when its monitored event queue changes state from empty to non-empty.

NO

The connection's list transition exit will not be called when its monitored event queue changes state. However, the vector index associated with the event queue will be updated to reflect the current empty/non-empty state information.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA

,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VECTORINDEX=*vectorindex*
Use this input parameter to specify the list notification vector index to be associated with the monitored event queue. If a start request completes successfully, the index identified by this vector index will reflect the event queue's state—either empty or non-empty. The timing as to when the index reflects the event queue's state is not always immediate; however, the index will eventually reflect the correct state of the event queue.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list notification vector index.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0852	**Equate Symbol:** IXLRSNCODENOLISTVECTOR **Meaning:** Program error. The request failed because no local vector for monitoring lists or the user's event queue exists for this connection. Either a list notification vector for monitoring lists or the user's event queue was not requested for this connection or the list notification vector has been deleted. **Action:** Either you are not connected to this structure or the VECTORLEN parameter was not specified when the IXLCONN was done for this structure.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0853	**Equate Symbol:** IXLRSNCODEINVLISTVINDEX **Meaning:** Program error. The list notification vector index (VECTORINDEX) specified with ACTION=START was not valid. This might be because the vector index you specified is greater than the number of vector entries in the list notification vector. **Action:**Correct the VECTORINDEX that you are using.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.

Table 54. Return and Reason Codes for IXLLIST REQUEST=MONITOR_EVENTQ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=MONITOR_LIST

Description

A MONITOR_LIST request allows you to start (ACTION=START) or stop (ACTION=STOP) monitoring a specified list (LISTNUM) for the presence of entries. When starting list monitoring, you must specify the list notification vector index (VECTORINDEX) to be associated with the LISTNUM list. List services uses the specified vector index to indicate whether the list is in the **empty** or **non-empty** state.

If you want to be notified when one of the lists for which list monitoring has been activated has changed state from empty to non-empty, you can specify DRIVEEXIT=YES. This option causes your list transition exit to be driven, informing you that a list that you monitor now has at least one entry on it. Your list transition exit must be identified on the LISTTRANEXIT parameter of the IXLCONN macro. To determine which of the monitored lists has changed state, you must use the IXLVECTR macro.

There are some considerations involving when you must stop list monitoring. If you have started list monitoring for a certain list with a certain vector index specified and you want to:

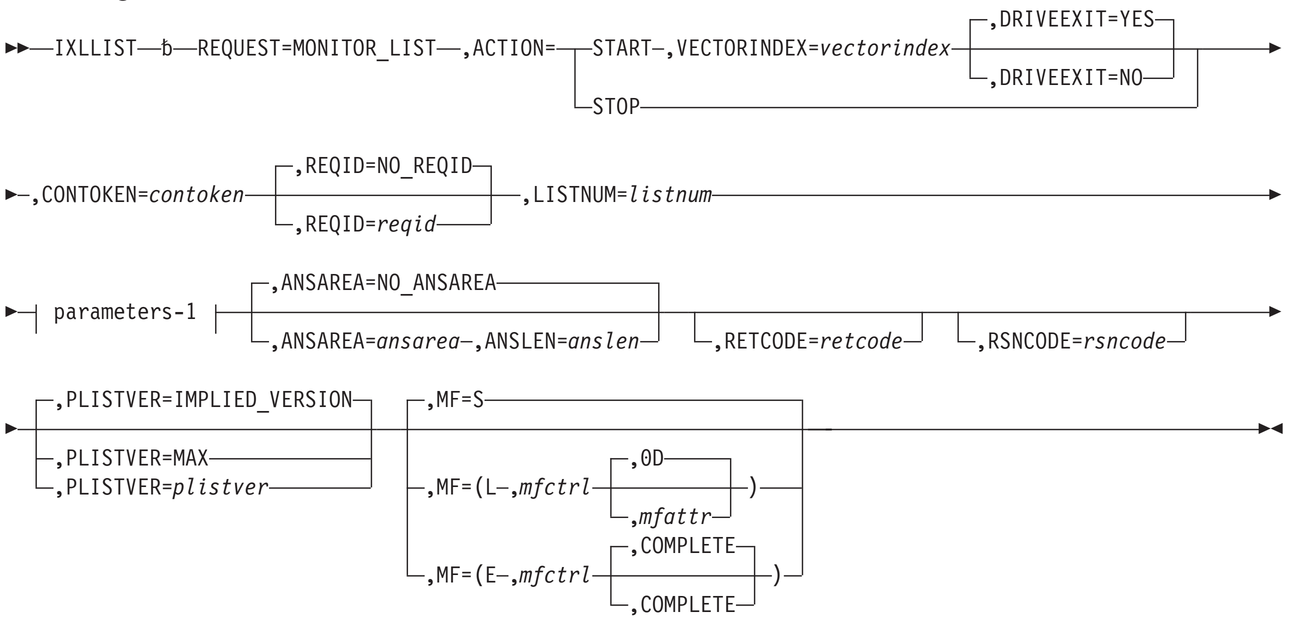
Monitor a different list with the same vector index
: You must stop list monitoring of the old list with the index before you start list monitoring of the new list with that index.

Monitor the same list with a different vector index
: Do not stop monitoring the list with the old vector index. Simply start monitoring the list with the new index; the new index will automatically replace the old.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=MONITOR_LIST is as follows:

main diagram



parameters-1

```
   ┌─,MODE=SYNCSUSPEND──────────────────────────┐
►►─┼────────────────────────────────────────────┼──────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb───────────────┤
   │                ┌─,REQDATA=NO_REQDATA─┐     │
   ├─,MODE=SYNCEXIT─┼─────────────────────┼─────┤
   │                └─,REQDATA=reqdata────┘     │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────┤
   │                 ┌─,REQDATA=NO_REQDATA─┐    │
   ├─,MODE=ASYNCEXIT─┼─────────────────────┼────┤
   │                 └─,REQDATA=reqdata────┘    │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────┤
   └─,MODE=ASYNCNORESPONSE──────────────────────┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, you may not specify ACTION=START.

Parameter Descriptions

The parameter descriptions for REQUEST=MONITOR_LIST are listed in alphabetical order. Default values are underlined:

REQUEST=MONITOR_LIST
Use this input parameter to specify that monitoring of the list specified by LISTNUM be started or stopped.

,ACTION=START
,ACTION=STOP
Use this input parameter to specify whether list monitoring is to be started or stopped.

START
Monitoring of the LISTNUM list is started.

STOP
Monitoring of the LISTNUM list is stopped.

Note: You cannot code ACTION=START with MODE=ASYNCNORESPONSE.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request with ACTION=START, the answer area contains the number of allocated entries or elements on the processed list (field LAALISTCNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DRIVEEXIT=YES
,DRIVEEXIT=NO

Use this input parameter to specify whether list services should drive the CONTOKEN connection's list transition exit when the state of the LISTNUM list changes state from empty to non-empty. The list transition exit informs the connection that one of the lists being monitored has changed state. To determine which of the monitored lists has changed state, use the IXLVECTR macro.

YES

The connection's list transition exit will be called when a monitored list changes state from empty to non-empty.

NO

The connection's list transition exit will not be called when a monitored list changes state. However, the vector index associated with the list will be updated to reflect the current empty/non-empty state information.

,LISTNUM=*listnum*

Use this input parameter to specify the number of the list you wish to monitor.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a descriptions of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,VECTORINDEX=*vectorindex*

Use this input parameter to specify the list notification vector index to be associated with the list specified by LISTNUM. If a start request completes successfully, the index identified by this vector index reflects the list's state—either empty or non-empty.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the list notification vector index.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection’s complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller’s primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0852	**Equate Symbol:** IXLRSNCODENOLISTVECTOR **Meaning:** Program error. The request failed because no local vector for monitoring lists or the user's event queue exists for this connection. Either a list notification vector for monitoring lists or the user's event queue was not requested for this connection or the list notification vector has been deleted. **Action:** Either you are not connected to this structure or the VECTORLEN parameter was not specified when the IXLCONN was done for this structure.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0853	**Equate Symbol:** IXLRSNCODEINVLISTVINDEX **Meaning:** Program error. The specified list notification vector index (VECTORINDEX) was not valid. This might be because the vector index you specified is greater than the number of vector entries in the list notification vector. **Action:**Verify the list being accessed. Change the vector index to a smaller value, or use the IXLVECTR macro to increase the size of the list notification vector. The number of vector indexes is specified on the VECTORLEN parameter for the IXLCONN macro. This value may also be changed by the IXLVECTR macro. The vector entries are numbered from 0 to n-1 where n is the number of entries.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.

Table 55. Return and Reason Codes for IXLLIST REQUEST=MONITOR_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=MONITOR_SUBLIST

Description

A MONITOR_SUBLIST request allows you to start (ACTION=START) or stop (ACTION=STOP) monitoring a designated sublist within the list structure. The sublist is identified by list number (LISTNUM) and entry key (ENTRYKEY). To accomplish sublist monitoring, you must specify when you connect to the list structure that you have a local vector associated with your connection to the structure (VECTORLEN keyword) and that the structure supports keyed entries (REFOPTION=KEY).

While sublist monitoring is in effect, the system will queue or withdraw event monitor controls (EMCs) on your event queue to indicate the empty or nonempty state of the sublist. Use sublist monitoring in conjunction with event queue monitoring to process state transitions for monitored sublists.

A MONITOR_SUBLIST request can be issued only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. MONITOR_SUBLIST requests issued for a list structure allocated in a coupling facility of CFLEVEL=0, 1, or 2 will fail.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=MONITOR_SUBLIST is as follows:

main diagram

```
►►──IXLLIST──␢──REQUEST=MONITOR_SUBLIST──,ACTION=─┬─START─,UNC=unc─┬─,CONTOKEN=contoken──────────────►
                                                  └─STOP───────────┘


   ┌─,REQID=NO_REQID─┐
►──┼─────────────────┼─,LISTNUM=listnum──,ENTRYKEY=entrykey─────────────────────────────────────────►
   └─,REQID=reqid────┘

                         ┌─,ANSAREA=NO_ANSAREA───────────────┐
►──┤ parameters-1 ├──────┼───────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬──►
                         └─,ANSAREA=ansarea─,ANSLEN=anslen───┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S─────────────────────────────────────┐
►──┼───────────────────────────┼───┼───────────────────────────────────────────┼──────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤   │                      ┌─,0D───────┐        │
   └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl───────┼───────────┼─)──────┤
                                   │                      └─,mfattr───┘        │
                                   │                      ┌─,COMPLETE─┐        │
                                   └─,MF=(E─,mfctrl───────┼───────────┼──)─────┘
                                                          └─,COMPLETE─┘
```

parameters-1

```
   ┌─,MODE=SYNCSUSPEND────────────────────────┐
►►─┼──────────────────────────────────────────┼─────────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb─────────────┤
   │                  ┌─,REQDATA=NO_REQDATA─┐ │
   ├─,MODE=SYNCEXIT───┼─────────────────────┼─┤
   │                  └─,REQDATA=reqdata────┘ │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken───────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb────────────┤
   │                   ┌─,REQDATA=NO_REQDATA─┐│
   ├─,MODE=ASYNCEXIT───┼─────────────────────┼┤
   │                   └─,REQDATA=reqdata────┘│
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken──────┤
   └─,MODE=ASYNCNORESPONSE────────────────────┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=MONITOR_SUBLIST are listed in alphabetical order. Default values are underlined:

REQUEST=MONITOR_SUBLIST
Use this input parameter to specify that monitoring of the sublist specified by LISTNUM and ENTRYKEY be started or stopped.

,ACTION=START
,ACTION=STOP
Use this input parameter to specify whether sublist monitoring is to be started or stopped.

START
Start monitoring the sublist identified by LISTNUM and ENTRYKEY or restart monitoring with new parameters.

STOP
Stop monitoring the sublist identified by LISTNUM and ENTRYKEY.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request with ACTION=START, the answer area contains the number of event monitor controls in use (field LAAMNSL_EMCCNT), the maximum number of event monitor controls for the structure (field LAAMNSL_MAXEMCCNT), and the state (empty or non-empty) of the sublist (field LAAMNSL_ENTRYQUEUED).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,ENTRYKEY=*entrykey*

Use this input parameter to specify the list entry key to be used in conjunction with LISTNUM to designate the sublist for which monitoring is to be started or stopped.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key of the sublist.

,LISTNUM=*listnum*

Use this input parameter to specify the number of the list containing the sublist you wish to monitor.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is

SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,UNC=*unc*

Use this input parameter to specify the user notification control information that represents the user's monitoring interest in the designated sublist. The user notification controls are contained in the event monitor controls (EMC) which is queued to the user's event queue when the monitored sublist becomes nonempty.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the user notification controls.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection’s complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0852	**Equate Symbol:** IXLRSNCODENOLISTVECTOR **Meaning:** Program error. The request failed because no local vector for monitoring lists or the user's event queue exists for this connection. Either a list notification vector for monitoring lists or the user's event queue was not requested for this connection or the list notification vector has been deleted. **Action:** Either you are not connected to this structure or the VECTORLEN parameter was not specified when the IXLCONN was done for this structure.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.

Table 56. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The request attempted to create a new event monitor controls (EMC) object, but the structure is full and cannot accommodate any more EMCs. **Action:** Determine why the structure is full. You should be monitoring the use of EMC objects (LAAMNSL_EMCCNT is the count of EMCs is use when sublist monitoring was established and LAAMNSL_MAXEMCCNT is the maximum number of EMCs for the structure.) Evaluate this data periodically to ensure structure resources are being used efficiently. You might be able to delete existing EMCs to free up space, or, if rebuild is allowed (IXLCONN macro, ALLOWREBLD parameter), rebuild the structure either to make it larger or with a changed EMCSTGPCT to allow more EMCs to be created.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=MONITOR_SUBLISTS

Description

A MONITOR_SUBLISTS request allows you to start monitoring a set of sublists in a list structure. Each sublist is designated by a list number and an entry key. You cannot stop monitoring the sublists with the MONITOR_SUBLISTS request. (Use the MONITOR_SUBLIST request to stop the monitoring of each individual sublist.)

While sublist monitoring is in effect, the system will queue or withdraw event monitor controls (EMCs) on your event queue to indicate the empty or nonempty state of the sublists. Use sublist monitoring in conjunction with event queue monitoring to process state transitions for monitored sublists.

Information about the sublists to be monitored is contained in the storage area specified by BUFFER or BUFLIST. The information for each sublist entry is mapped by the IXLYMSRI macro. The entries are indexed with the first entry starting at offset zero in the BUFFER or BUFLIST area. To designate the entries to be monitored, specify the starting and ending index numbers of the IXLYMSRI entries with the STARTINDEX and ENDINDEX keywords.

A MONITOR_SUBLISTS request can complete prematurely for the following reasons:

- The request has exceeded the model-dependent time-out criteria. The index of the next IXLYMSRI entry to be processed is returned in the answer area (ANSAREA).
- There is insufficient event monitor control space in the list structure to register as a sublist monitor. The index of the first unprocessed IXLYMSRI entry (the entry that experienced the failure) is returned in the answer area.
- The user has specified a list number that is not valid in an IXLYMSRI entry. The index of the failing IXLYMSRI entry is returned in the answer area.

Before restarting a MONITOR_SUBLISTS request that has completed prematurely, you should process the entries that were successfully returned. Determine how to restart based on the type of premature completion that occurred.

- In the case of model-dependent time-out you can update STARTINDEX to field LAAMNSLS_FAILINDEX in the answer area on a subsequent MONITOR_SUBLISTS request to finish processing the entries.
- In the case of insufficient EMC space you must take actions to relieve the space constraints before updating STARTINDEX on a subsequent MONITOR_SUBLISTS request.
- In the case of a list number that is not valid, you must correct the erroneous specification before updating STARTINDEX on a subsequent MONITOR_SUBLISTS request.

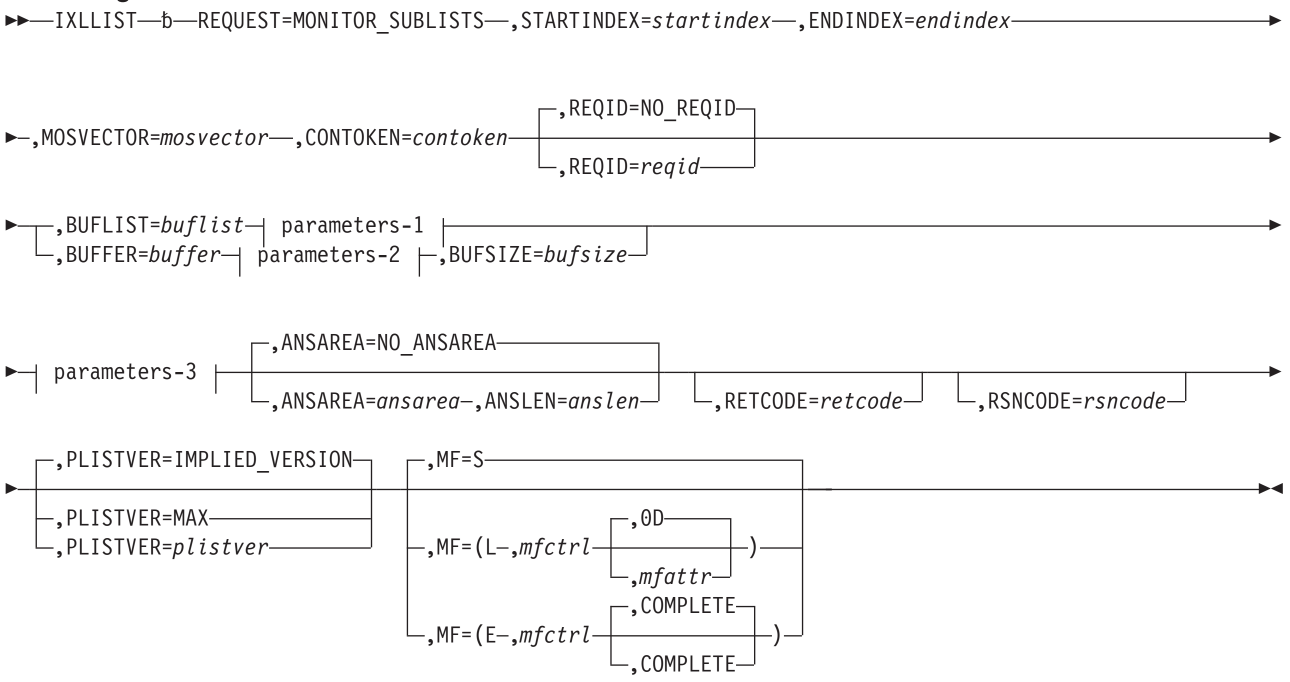
A MONITOR_SUBLISTS request can be issued only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. MONITOR_SUBLISTS requests issued for a list structure allocated in a coupling facility of CFLEVEL 0, 1, or 2 will fail.

Syntax Diagram

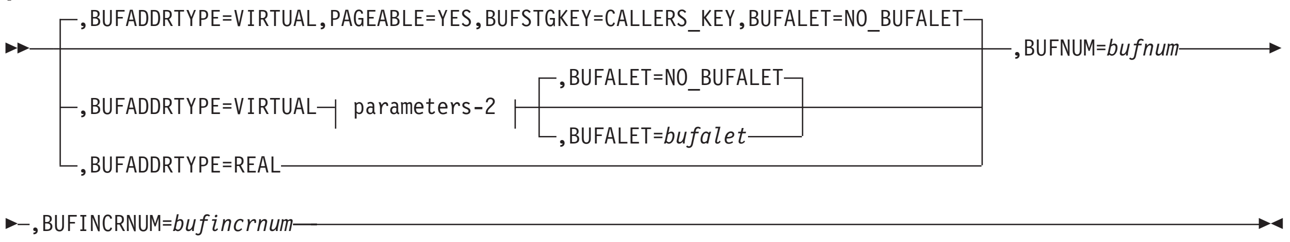
The syntax diagram for IXLLIST REQUEST=MONITOR_SUBLISTS is as follows:

IXLLIST Macro

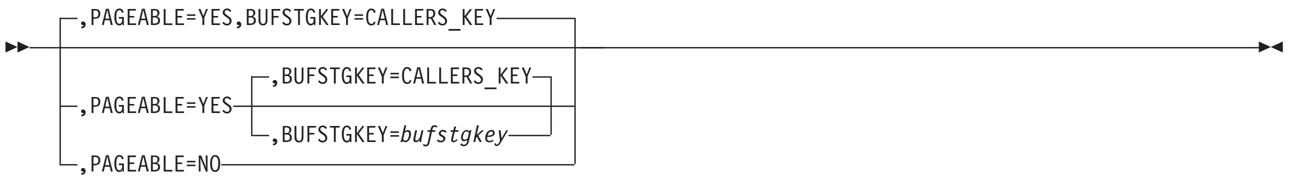
main diagram



parameters-1



parameters-2



parameters-3

```
   ┌─,MODE=SYNCSUSPEND──────────────────────────────┐
►►─┼────────────────────────────────────────────────┼──────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤
   │                 ┌─,REQDATA=NO_REQDATA─┐        │
   ├─,MODE=SYNCEXIT──┼─────────────────────┼────────┤
   │                 └─,REQDATA=reqdata────┘        │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
   │                  ┌─,REQDATA=NO_REQDATA─┐       │
   ├─,MODE=ASYNCEXIT──┼─────────────────────┼───────┤
   │                  └─,REQDATA=reqdata────┘       │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┤
   └─,MODE=ASYNCNORESPONSE──────────────────────────┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=MONITOR_SUBLISTS are listed in alphabetical order. Default values are underlined:

REQUEST=MONITOR_SUBLISTS
Use this input parameter to monitor a set of sublists that are identified in an input array of entries.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

When the request completes successfully or completes prematurely, the answer area contains the number of event monitor controls in use (field LAAMNSLS_EMCCNT) and the maximum number of event monitor controls for the structure (field LAAMNSLS_MAXEMCCNT). For a premature completion, the answer area also contains the index of the first unprocessed IXLYMSRI entry (field LAAMNSLS_FAILINDEX). This index can be specified (using the STARTINDEX parameter) on the next MONITOR_SUBLISTS request to finish processing the entries in the input array.

See *z/OS MVS Programming: Sysplex Services Guide* for a description of all the relevant IXLYLAA fields for this request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information returned by the request.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this input parameter to specify a buffer area to hold an array of entries that identify the sublists to be monitored. Each entry is mapped by the IXLYMSRI macro.

You can define the buffer size to be a total of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
  - Does not start below storage address 512.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.
  - Does not start below storage address 512.

See the BUFSIZE parameter description for defining the size of the buffer.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a size of BUFSIZE) that contains the array of sublist identifiers.

,BUFINCRNUM
Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*
Use this input parameter to specify a list of buffers to hold the array of entries that identify the sublists to be monitored.

BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:
- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:

- The left (high-order) four bytes are reserved and
- The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.
- Not start below storage address 512.

Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single buffer even if the buffers are not contiguous.

See the BUFNUM and BUFINCRNUM parameter descriptions for specifying the number and size of buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no sublists are to be monitored.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller’s PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,ENDINDEX=*endindex*

Use this input parameter to specify the index number of the last IXLYMSRI entry to be processed. The valid range is from the STARTINDEX value to 1024.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the index of the last entry to be processed.

,MF=S
,MF=(L,*mfctrl***)**

,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,MOSVECTOR=*mosvector*
Use this output parameter to specify a 128-byte field to contain the bit string representing the monitored object state (empty or nonempty) of each sublist at the time the MONITOR_SUBLISTS request was processed. The MOSVECTOR bits correspond one-to-one with the IXLYMSRI entries that were passed as input in BUFFER or BUFLIST. Only the bits corresponding to the IXLYMSRI entries that were actually processed on the current request (that is, those between STARTINDEX and ENDINDEX for a request that completed successfully, or between STARTINDEX and the returned index of the first unprocessed entry minus one for requests that completed prematurely) will contain valid monitored object state information for the sublists designated by the corresponding IXLYMSRI entries. Bits in the MOSVECTOR that lie outside the valid range are not meaningful.

When a bit in the valid range is on, the sublist designated by the corresponding IXLYMSRI entry was nonempty at the time the request was processed. When a bit in the valid range is off, the sublist designated by the corresponding IXLYMSRI entry was empty at the time the request was processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field to contain the bit string indicating the monitored object state of each sublist for which monitoring was requested.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,STARTINDEX=*startindex*
Use this input parameter to specify the index number of the first IXLYMSRI entry to be processed. The valid range is from 1 to the ENDINDEX value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the index of the first entry to be processed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

IXLLIST Macro

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of event monitor controls in use (field LAAMNSLS_EMCCNT) • The maximum number of event monitor controls for the structure (field LAAMNSLS_MAXEMCCNT) • The index of the next IXLYMSRI entry to be processed (field LAAMNSLS_FAILINDEX). **Action:** After processing all returned data, reissue the request beginning with the first unprocessed entry. For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The value specified for either STARTINDEX or ENDINDEX was not valid. No entries were processed. The index of the first entry that was not processed is returned in the answer area. **Action:** Ensure that the STARTINDEX and ENDINDEX values are valid and resubmit the request.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0852	**Equate Symbol:** IXLRSNCODENOLISTVECTOR **Meaning:** Program error. The request failed because no local vector for monitoring lists or the user’s event queue exists for this connection. Either a list notification vector for monitoring lists or the user’s event queue was not requested for this connection or the list notification vector has been deleted. **Action:** Either you are not connected to this structure or the VECTORLEN parameter was not specified when the IXLCONN was done for this structure.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size, and reissue the request.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • The requirements for BUFLIST or BUFFER. • Buffer pointer(s) in the BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx0880	**Equate Symbol:** IXLRSNCODEBADMOSVECTOR **Meaning:** Program error. The storage area specified by MOSVECTOR is not addressable. **Action:** Ensure that: • The correct storage area address was used. • If you are running in AR-mode and the MOSVECTOR was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The request attempted to create a new event monitor controls (EMC) object, but the structure is full and cannot accomodate any more EMCs. **Action:** Determine why the structure is full. You should be monitoring the usage of the structure every time a REQUEST=MONITOR_SUBLISTS is done (LAAMNSLS_EMCCNT is the total count of EMCs in use in the list structure and LAAMNSLS_MAXEMCCNT is the maximum number of EMCs for the list structure.). This data should be evaluated periodically to ensure structure resources are being used efficiently. Field LAAMNSLS_FAILINDEX contains the index of the entry in IXLYMSRI that experienced the structure full condition. IXLYMSRI entries prior to LAAMNSLS_FAILINDEX were processed successfully.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.

Table 57. Return and Reason Codes for IXLLIST REQUEST=MONITOR_SUBLISTS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=MOVE

Description

A MOVE request allows you to perform one of the following types of operations:

- Move an existing list entry from one list to another list or within the same list. In addition to moving the entry, you can also read or update that entry's data.
- Create a new list entry (if the entry does not exist on a specified list), place it on either the same or another list, and write data to it.

Use the DATAOPER parameter to specify the operation to be performed:

- **DATAOPER=NONE** Only the existing entry is moved.
- **DATAOPER=READ** The existing entry is moved, and its data is read.
- **DATAOPER=WRITE**
  - ENTRYTYPE=OLD The existing entry is moved, and its data is updated.
  - ENTRYTYPE=ANY If the designated entry exists, it is moved and its data is updated. If the entry does not exist, a new entry is created and data is written to it.

If this request creates a new entry (ENTRYTYPE=ANY must be specified) you must provide the list number (MOVETOLIST) on which the new entry will be placed. You can also provide additional identification for the new entry by specifying one of the following:

- Entry name (ENTRYNAME)
- Entry key (ENTRYKEY)

Note: You cannot assign an entry identifier (ENTRYID) to a new entry; the system assigns entry identifiers to all new entries.

A MOVE request allows you to read (DATAOPER=READ) or write (DATAOPER=WRITE) entry data and/or adjunct data. Entry data is read from or written to the **data entry** segment of an entry, while the adjunct data is read from or written to the **adjunct data area** segment. The following describes from where each type of data is written (if DATAOPER=WRITE) or where each type of data is returned (if DATAOPER=READ):

- A buffer (BUFFER) or list of buffers (BUFLIST) holds entry data. If neither BUFFER nor BUFLIST is specified, no entry data is read or written.
- An adjunct area (ADJAREA) holds adjunct data. If ADJAREA is not specified, no adjunct data is read or written.

If you specify DATAOPER=WRITE, you can assign the number of data elements comprising the data entry segment of the existing or new entry by specifying ELEMNUM.

Additionally, you can perform locking functions with a MOVE request by specifying LOCKINDEX. The lock entry designated by LOCKINDEX will be operated on as specified by LOCKOPER.

With a coupling facility of CFLEVEL=1 or higher, the following additional functions are supported for a MOVE request:

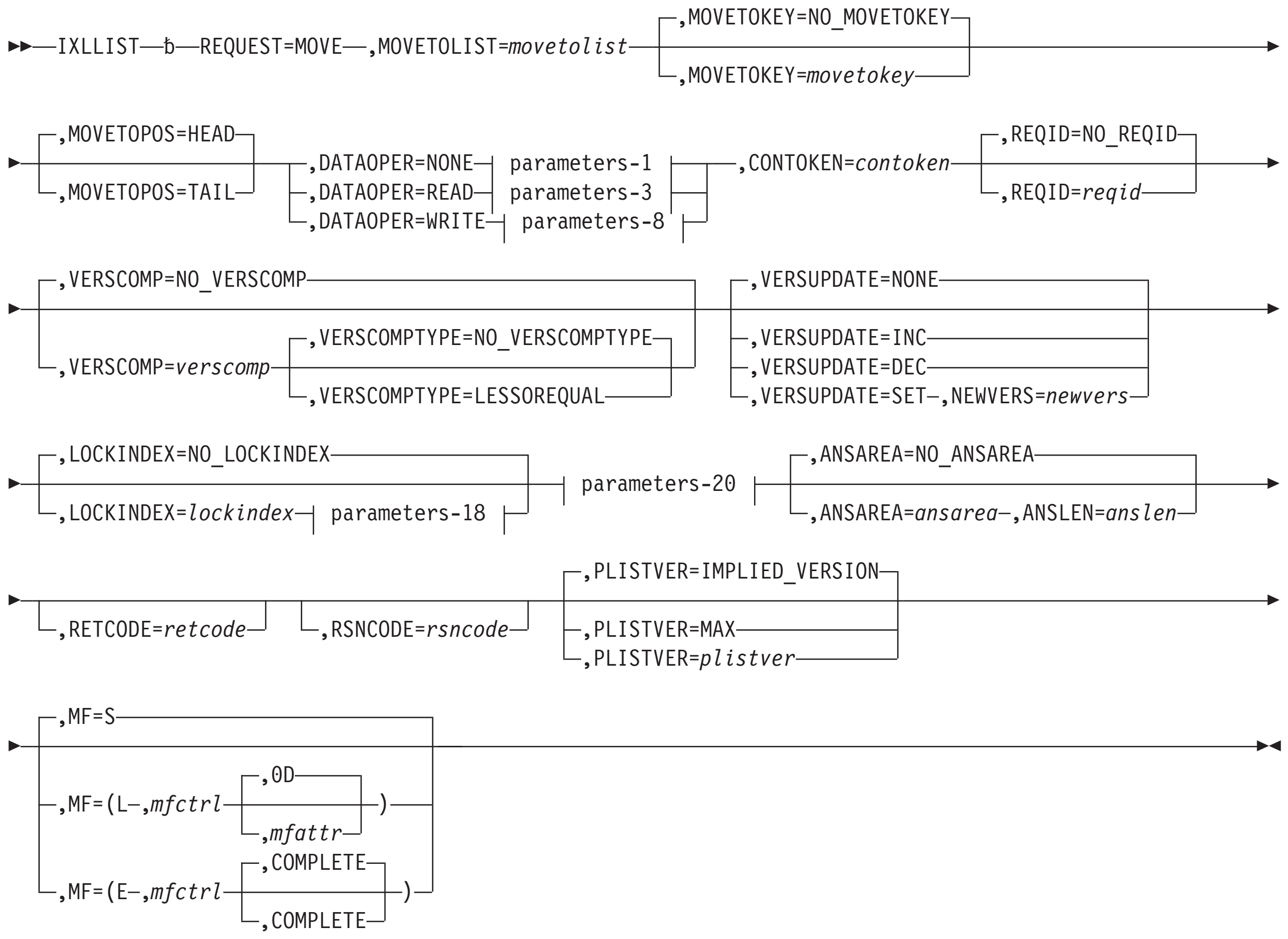
- You can specify that a list entry be moved only if the list authority value for the list associated with the entry is of a certain value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. If the request is successful, you also can update the list authority value to a new value that you specify with NEWAUTH.
- Version number comparison is enhanced to allow for an additional equal-or-less-than-equal comparison operation.
- The system can assign an entry key value automatically from a list control list key value.

- There are additional list cursor options. You can update the list cursor conditionally, and you can set the list cursor to point to the current entry instead of the previous or next entry.

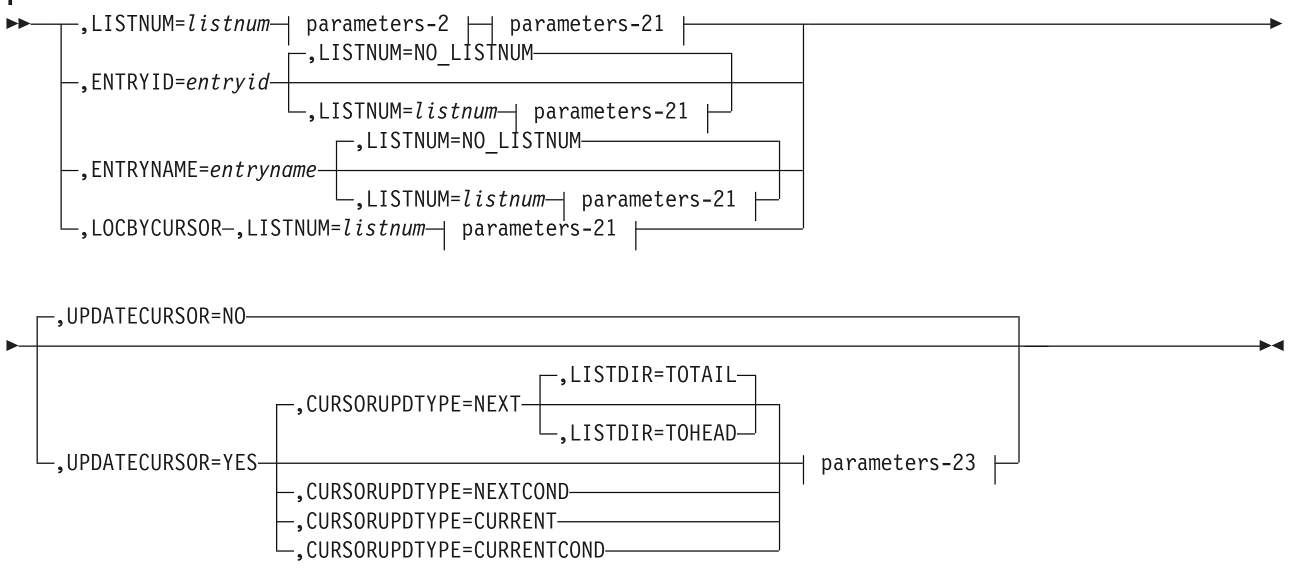
Syntax Diagram

The syntax diagram for IXLLIST REQUEST=MOVE is as follows:

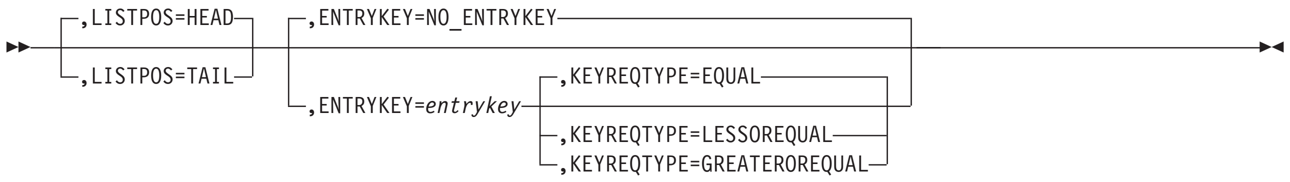
main diagram



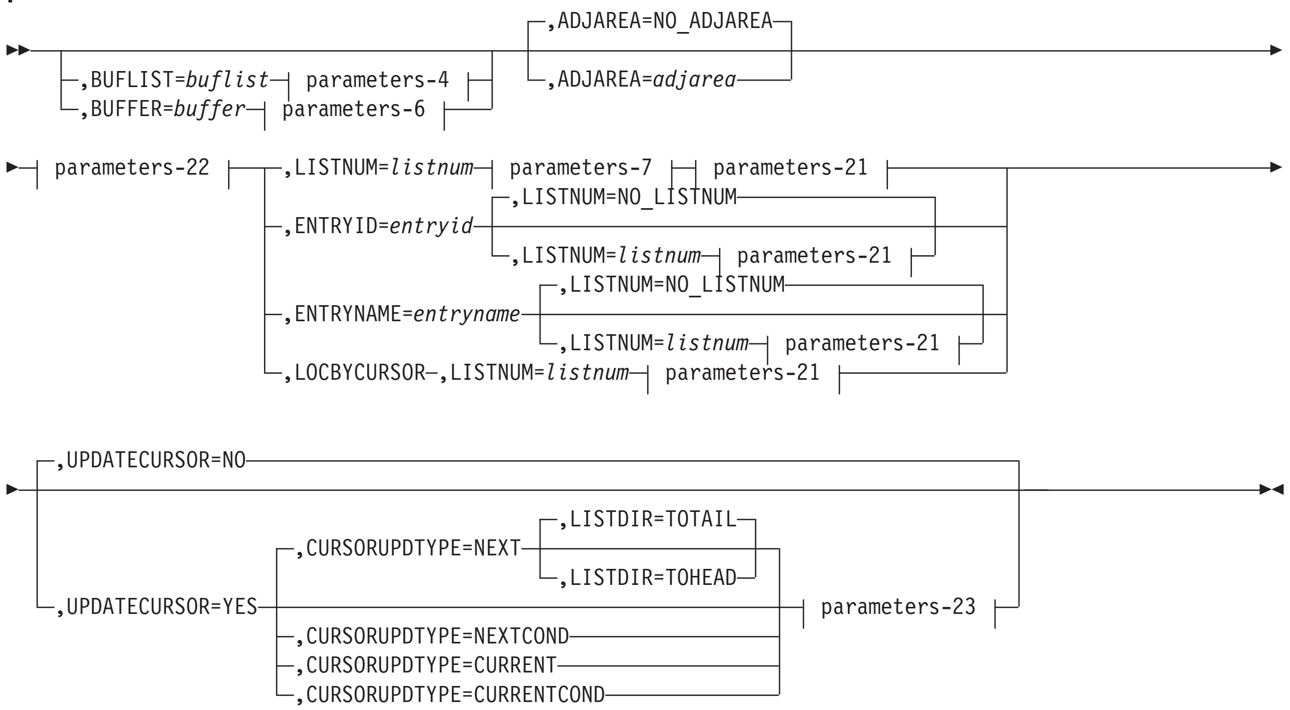
parameters-1



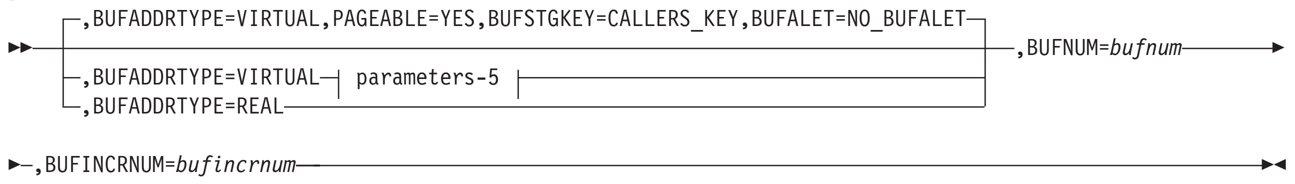
parameters-2



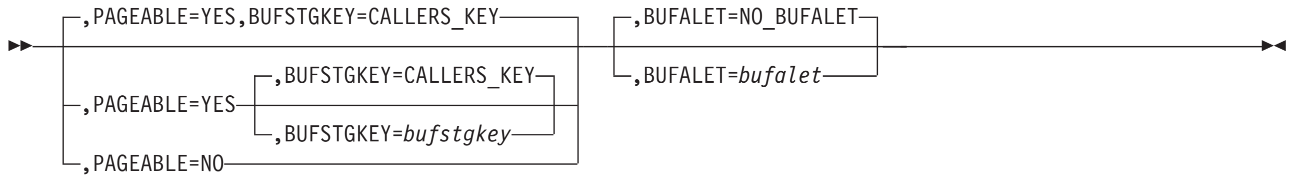
parameters-3



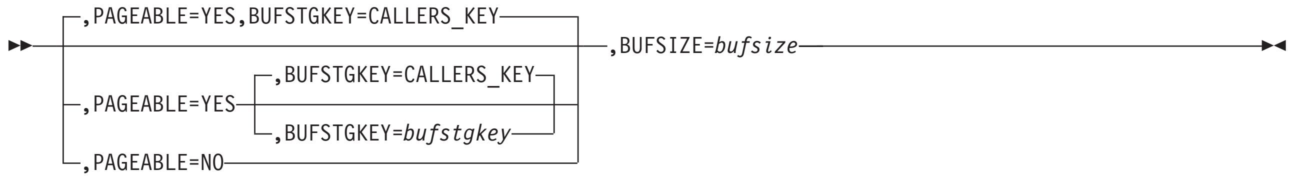
parameters-4



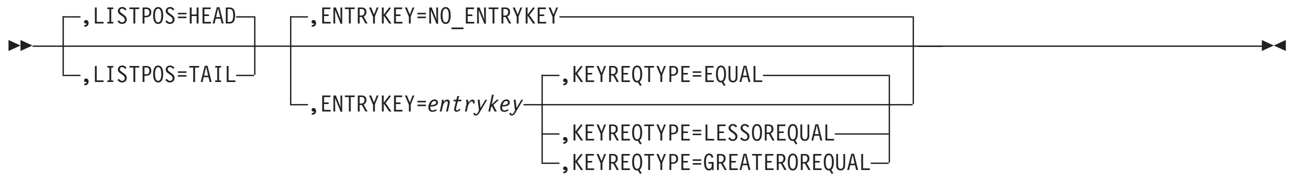
parameters-5



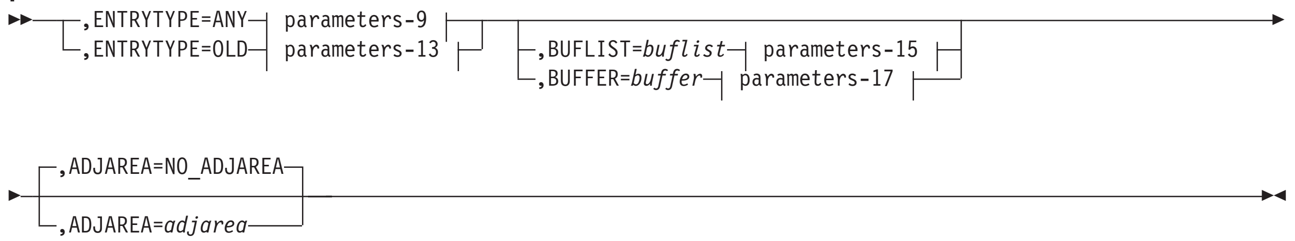
parameters-6



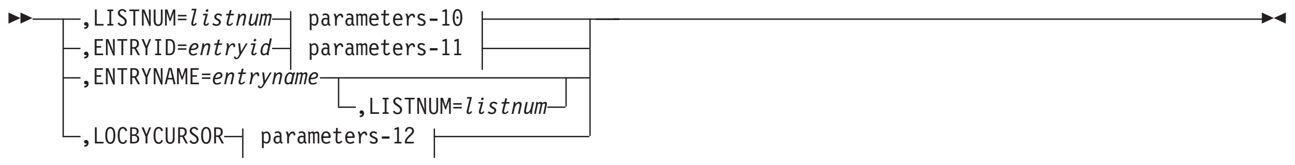
parameters-7



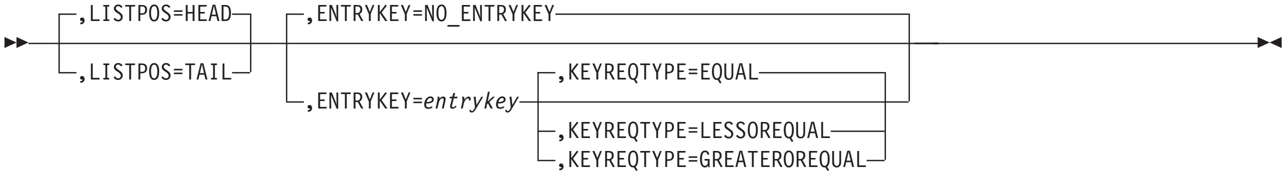
parameters-8



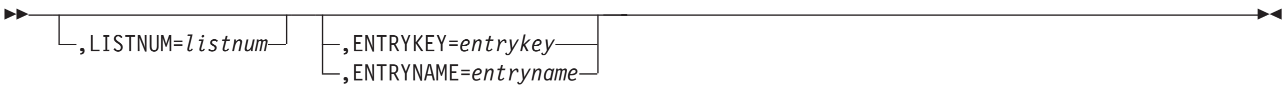
parameters-9



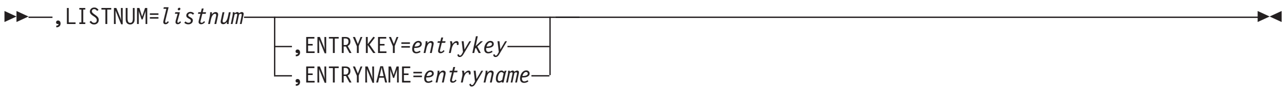
parameters-10



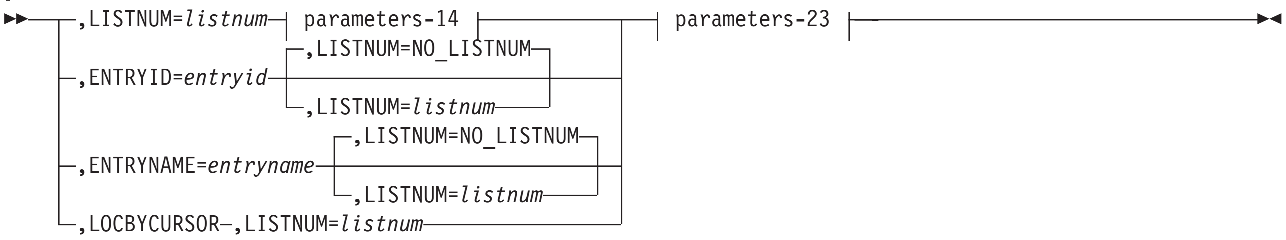
parameters-11



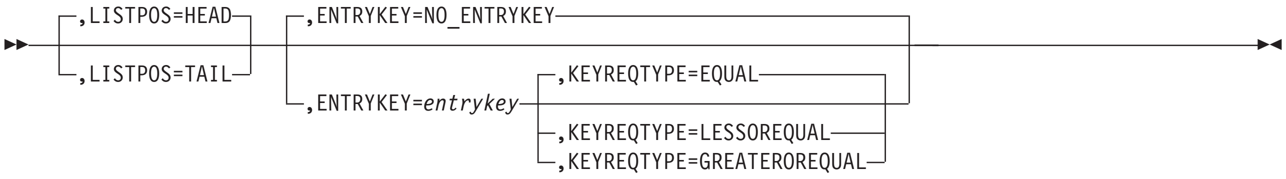
parameters-12



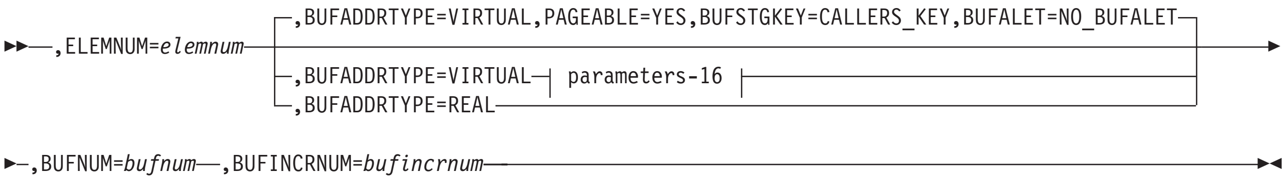
parameters-13



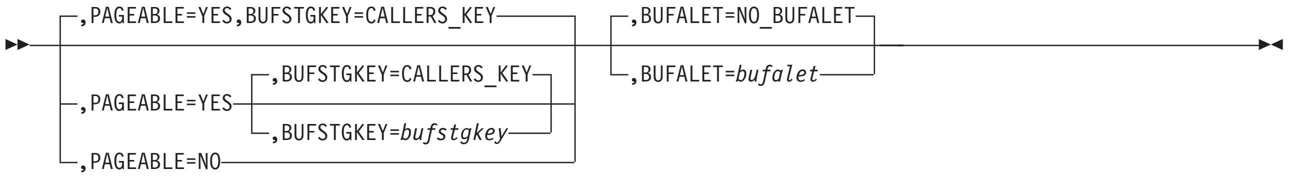
parameters-14



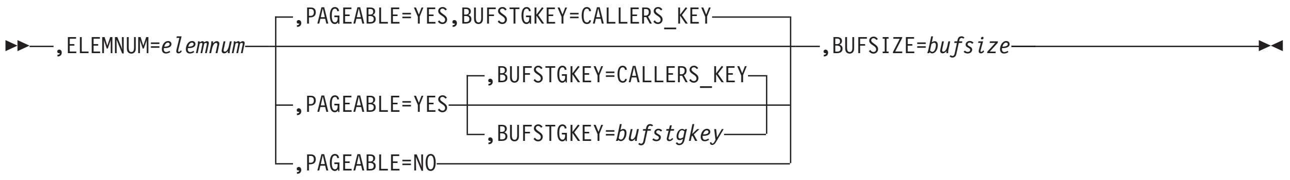
parameters-15



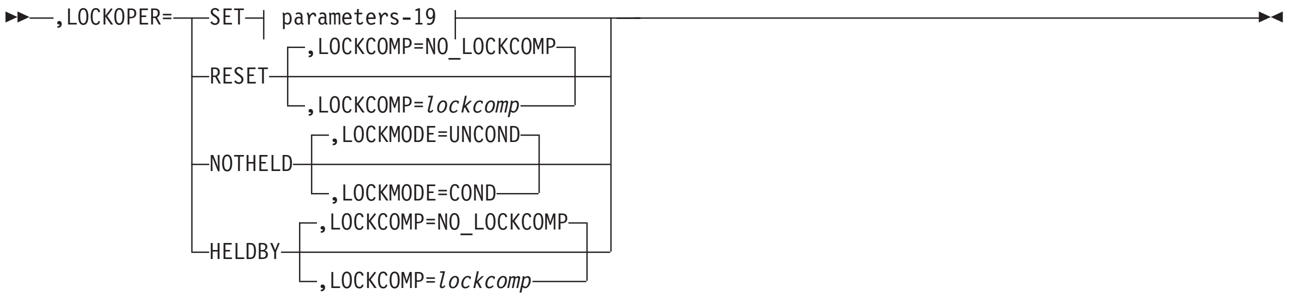
parameters-16



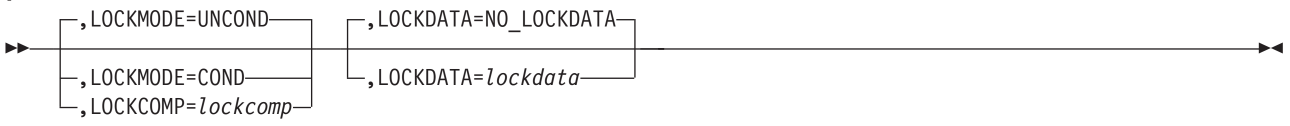
parameters-17



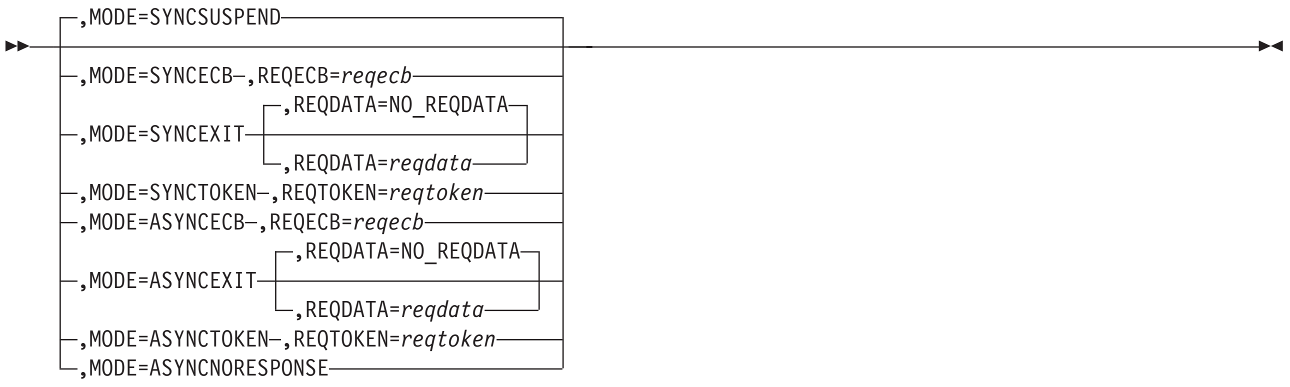
parameters-18



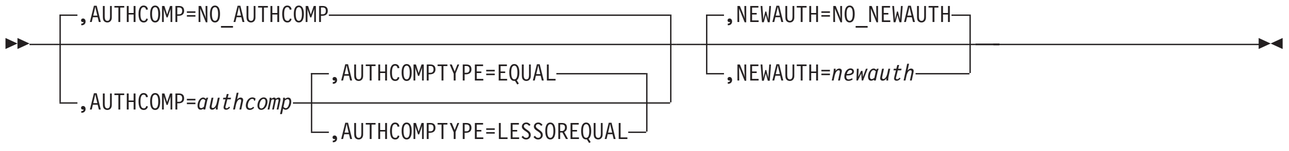
parameters-19



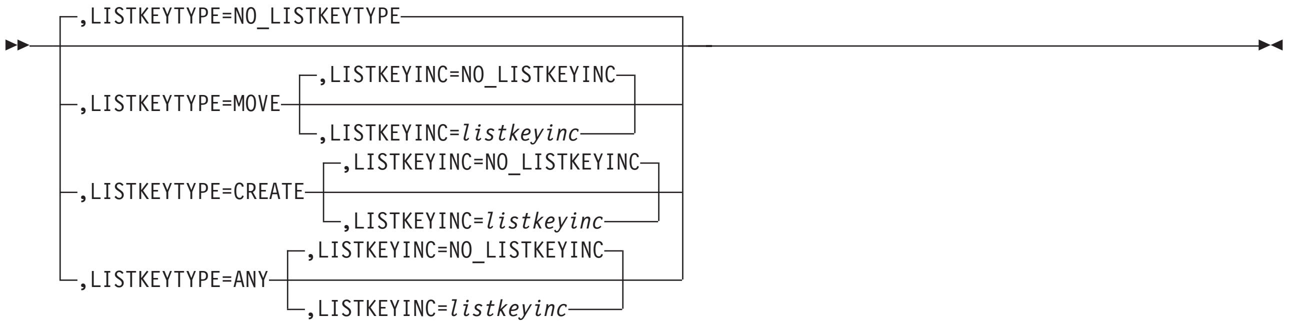
parameters-20



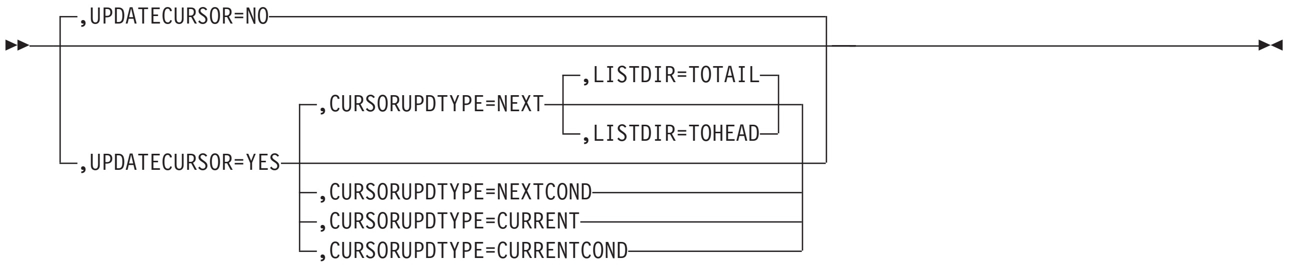
parameters-21



parameters-22



parameters-23



Notes:

1. In the main diagram, if DATAOPER is not specified, DATAOPER=NONE is the default and you must code the required parameters shown in the | parameters-1 | fragment.
2. In the | parameters-3 | fragment, one of the following must be specified:
   - BUFLIST=*buflist*
   - BUFFER=*buffer*
   - ADJAREA=*adjarea*

   In addition, ADJAREA=*adjarea* can be specified with either BUFLIST=*buflist* or BUFFER=*buffer*.
3. In the | parameters-8 | fragment, if ENTRYTYPE is not specified, ENTRYTYPE=ANY is the default and you must code the required parameters specified in the | parameters-9 | fragment.
4. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA= *ansarea* and ANSLEN=*anslen* are required.
5. If MODE=ASYNCNORESPONSE is specified, BUFFER, BUFLIST, and LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=MOVE are listed in alphabetical order. Default values are underlined:

REQUEST=MOVE
Use this input parameter to move an entry.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*

Use this input parameter to specify a storage area to contain the adjunct data that is read from or written to an entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area that contains or will contain the adjunct data.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion (for a synchronous request: RC=0, for an asynchronous request: ECB is posted, IXLFCOMP indicates completion, or your complete exit ran) of the request, the following information is returned to the answer area:

- List entry controls of the existing or new entry (field LAALCTL).
- The number of list entries or elements residing on the list (field LAALISTCNT).
- The total number of allocated entries in the structure (field LAATOTALCNT).
- The total number of allocated elements in the structure (field LAATOTALELECNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned by the request.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL

The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL

The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output or input parameter to hold entry data to be read from or written to the designated entry. The purpose of the BUFFER area depends on the DATAOPER value you specify:

- If you specify DATAOPER=READ, the buffers hold output data—entry data that was read from an existing entry.
- If you specify DATAOPER=WRITE, the buffers hold input data—entry data to be written to an existing or new entry.

You can define the buffer size to be a total size of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

See the BUFSIZE parameter description for defining the size of the buffer.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry data.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this output or input parameter to specify a list of buffers to hold entry data to be read from or written to the designated entry. The purpose of the BUFLIST buffers depend on the DATAOPER value you specify:

- If you specify DATAOPER=READ, the buffers hold output data—entry data that was read from an existing entry.
- If you specify DATAOPER=WRITE, the buffers hold input data—entry data to be written to an existing or new entry.

BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:

- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:

- Reside in either the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

  Note: The buffers do not have to be contiguous in storage. (List services treats BUFLIST buffers as a single, contiguous buffer, even if the buffers are not contiguous.)

  See the BUFNUM and BUFINCRNUM parameter descriptions for defining the number and size of buffers.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers or written to the list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CURSORUPDTYPE=NEXT
,CURSORUPDTYPE=NEXTCOND
,CURSORUPDTYPE=CURRENT
,CURSORUPDTYPE=CURRENTCOND

Use this input parameter to specify how the list cursor is to be updated when UPDATECURSOR=YES is specified.

Note: The NEXTCOND, CURRENT, and CURRENTCOND values are valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

CURSORUPDTYPE=NEXT

Update the list cursor to point to the list entry before or after the target entry.

- For MOVE requests that specify DATAOPER=WRITE and ENTRYTYPE=ANY and that result in the creation of a new entry, the cursor for the list specified by MOVETOLIST is updated. The direction of the cursor update depends on the value specified for MOVETOPOS.
- For all other requests, the cursor for the source list is updated. The direction of the cursor update depends on the value specified for LISTPOS or LISTDIR.

CURSORUPDTYPE=NEXTCOND

Update the list cursor to point to the list entry before or after the target entry only if the cursor points to the target list entry, and that list entry is moved or deleted.

Set the list cursor direction with SETCURSOR on a WRITE_LCONTROLS request.

The list cursor will be reset to binary zeros if either:

- The entry is the last entry on the list and the list cursor direction is set to a head-to-tail direction.
- The entry is the first entry on the list and the list cursor is set to a tail-to-head direction.

CURSORUPDTYPE=CURRENT

Set the list cursor to point to the list entry processed for the request.

If the list entry is deleted or moved to another list, then the list cursor will be reset to binary zeros.

CURSORUPDTYPE=CURRENTCOND

Set the list cursor to point to the list entry processed only if the list cursor value currently is zero and the target list entry is not deleted or moved to another list.

If the list entry is deleted or moved to another list, then the list cursor will remain binary zeros.

,DATAOPER=NONE
,DATAOPER=READ
,DATAOPER=WRITE

Use this input parameter to specify an operation to be performed on the designated entry.

NONE

The existing entry is moved; no additional operation is performed.

READ
The existing entry is moved, and its entry data and/or adjunct data is read:

- If you specified BUFFER or BUFLIST, the entry data is read into whichever buffer area you have specified.
- If you specified ADJAREA, the adjunct data is read into the adjunct area.

WRITE
The existing entry is moved, and its entry data and/or adjunct data is updated. If this request creates a new entry, the entry data and/or adjunct data is written to the new entry:

- If you specified BUFFER or BUFLIST, the entry data is written to the existing or new entry.
- If you specified ADJAREA, the adjunct data is written to the existing or new entry.

,ELEMNUM=*elemnum*
Use this input parameter to specify the number of elements to be allocated to the existing or new data entry. Valid ELEMNUM values are from zero to the MAXELEMNUM value that was specified on the IXLCONN macro.

Considerations for specifying ELEMNUM are:

- If this request creates a new entry, the number of elements is set to the ELEMNUM value.
- If the entry already exists, the number of elements is updated to the ELEMNUM value specified.

Note: If the data from the buffers exceeds the amount of space in the elements, the data will be truncated. If the data from the buffers is less than the amount of space in the elements, the remaining space will be padded with binary zeros.

If you specify an ELEMNUM value of zero:

- No entry data will be written to the new entry.
- Any entry data in the existing entry will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of elements.

,ENTRYID=*entryid*
Use this input parameter to designate the entry identifier of an existing entry to be moved. ENTRYID applies only to locating an existing entry; it does not determine the identifier of a new entry should one be created.

If an entry with this identifier does not exist (and ENTRYTYPE=ANY is specified), a new entry is created on the MOVETOLIST list with a unique entry identifier assigned by list services. (The entry identifier you specify will be ignored.)

When ENTRYID is specified with ENTRYNAME, ENTRYKEY, or LISTPOS, list services uses only ENTRYID to determine if the entry already exists.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=NO_ENTRYKEY
,ENTRYKEY=*entrykey*
Use this input parameter to either:

- Designate the entry key of an existing entry to be moved.
- Assign an entry key to a new entry to be created, if MOVETOKEY is not specified. If MOVETOKEY is specified, the MOVETOKEY key is assigned to the new entry.

The existing entry must reside on the LISTNUM list. The existing entry will be placed on the MOVETOLIST list. If a new entry is created it will be placed on the MOVETOLIST list.

If DATAOPER=NONE or DATAOPER=READ is specified, the specified entry key designates an existing entry.

If DATAOPER=WRITE is specified, the specified entry key designates an existing entry or can be assigned to a new entry. See *z/OS MVS Programming: Sysplex Services Guide* for information about how entry keys are assigned to existing or created entries.

Specify ENTRYKEY only for structures that support keyed entries.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*

Use this input parameter to either:

- Designate an existing entry to be moved.
- Assign an entry name to a new entry to be created.

Both existing and new entries will be placed on the MOVETOLIST list.

If DATAOPER=NONE or DATAOPER=READ, the specified entry name designates an existing entry.

If DATAOPER=WRITE is specified, whether the specified entry name designates an existing entry or is assigned to a new entry depends on the following:

- If either of the following is true, the specified entry name designates an existing entry:
  - ENTRYTYPE=OLD, and the entry already exists.
  - ENTRYTYPE=ANY, neither ENTRYID nor LOCBYCURSOR is also specified, and the entry already exists.
  - If ENTRYTYPE=ANY and the entry does not already exist, a new entry is created with the ENTRYNAME name assigned to it.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,ENTRYTYPE=ANY
,ENTRYTYPE=OLD

Use this input parameter to specify whether you want to either:

- Update an existing entry or create a new entry
- Update an existing entry

ANY
Either update an existing entry or create a new entry. If the entry exists, its data is replaced with the new data. If the entry does not exist, one will be created with the new data.

OLD
Update an existing entry. The entry must already exist, or the request fails.

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL

Use this input parameter to specify how, if at all, the entry key of the existing entry to be moved can differ from the entry key specified by ENTRYKEY. KEYREQTYPE applies only to locating an existing entry; it does not determine the key of a new entry should one be created.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

Notes:

1. When no entries on the list meet the requirements of the KEYREQTYPE, and a new entry cannot be created (TYPE=OLD was specified), the request will be failed with a return code X'8' and reason code IXLRSNCODENOENTRY, or a new list entry will be allocated, depending on the other options specified.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, then the entry with an entry key value closest to the value specified will be selected. When multiple entries have the same entry key value, LISTPOS is used to resolve whether the first or last entry with the entry key value is selected.

,LISTDIR=TOTAIL
,LISTDIR=TOHEAD
Use this input parameter with UPDATECURSOR to specify the direction in which the list cursor is moved as a result of this request. See the UPDATECURSOR parameter for a full description of LISTDIR.

,LISTKEYINC=NO_LISTKEYINC
,LISTKEYINC=*listkeyinc*
Use this input parameter to specify a value to be added to the list key after the entry key is set to the list key value. If the entry key is not set to the list key value, the list key value will not be changed. If the result of adding the value specified by LISTKEYINC to the list key value is greater than the maximum list key value, the system fails the request.

Note: The LISTKEYINC parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value to be added to the list key after the entry key is set to the list key value.

,LISTKEYTYPE=NOLISTKEY
,LISTKEYTYPE=MOVE
,LISTKEYTYPE=ANY
Use this input parameter to specify when the designated entry key is to be set to the current list key value. You can set the entry key to the current list key value when you move the entry.

(Set the list key and maximum list key values with a WRITE_LCONTROLS request.)

Note: The LISTKEYTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1.

LISTKEYTYPE=NOLISTKEY
Do not set the entry key for the target list entry to the current list key value. If the list entry is moved and the list structure supports entries with keys, then the other parameters specified (such as ENTRYKEY) will determine the resultant entry key value.

LISTKEYTYPE=MOVE
Set the entry key for the designated list entry to the current list key value if the entry is moved by this request.

LISTKEYTYPE=ANY
Set the entry key for the designated list entry to the current list key value if the entry is moved or created by this request. (This parameter also applies when you WRITE an entry.)

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entry to be moved currently resides. Use LISTNUM in the following ways:

- With LISTDIR and/or ENTRYKEY to designate an existing entry to be moved.
- With ENTRYID or ENTRYNAME to:
  - Check if the entry exists on the list (when ENTRYTYPE=ANY) before proceeding with the request.
  - Confirm that the entry exists on the list (when ENTRYTYPE=OLD) before proceeding with the request.
- With LOCBYCURSOR to designate an existing entry to be moved.

To Code: Specify the RS-type name or address (using a 2 register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTPOS=HEAD
,LISTPOS=TAIL

Use this input parameter with LISTNUM to designate the existing entry to be moved. LISTPOS applies only to locating an existing entry; it does not determine where a new entry would be placed should one be created. (See the MOVETOPOS parameter for a description of where a new entry is placed.)

LISTPOS designates the entry at the HEAD or the TAIL of a list or sublist:

- If you specify ENTRYKEY to designate the entry and the list contains a sublist of one or more entries with the same key (or that satisfies the KEYREQTYPE criteria), then:
  - LISTPOS=HEAD designates the entry at the head of the sublist.
  - LISTPOS=TAIL designates the entry at the tail of the sublist.
- If you do not specify ENTRYKEY to designate the entry:
  - LISTPOS=HEAD designates the entry at the head of the list.
  - LISTPOS=TAIL designates the entry at the tail of the list.

,LOCBYCURSOR

Use this input parameter to designate an existing entry to be moved. The designated entry is the entry to which the LISTNUM list cursor is pointing.

Be aware that the list cursor could have been reset to zeros by a previous request that, for instance, deleted or moved the entry to which the list cursor points, or updated the list cursor after the last entry on the list had been processed. See *z/OS MVS Programming: Sysplex Services Guide* for more information about using the list cursor.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA

,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:

- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND

Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND

If the specified lock is held by another user, your request is either:

- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND

The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET

Set the lock.

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released or fails, depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

RESET

Reset the lock.

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

NOTHELD
The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released or your request fails, depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

HELDBY
Processing is determined by which connector is holding the lock.

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT

,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,MOVETOKEY=NO_MOVETOKEY
,MOVETOKEY=*movetokey*

Use this input parameter to assign a new key, and hence the position on the MOVETOLIST, for the existing entry being moved or the new entry being created. In the case of an existing entry, the existing entry's key will be updated to the MOVETOKEY key value. If MOVETOKEY is not specified, ENTRYKEY defines the key to use for the entry.

Notes:

1. If there is a sublist of one or more entries with a matching key on the list then the target position is the head or tail of the sublist, as specified by the MOVETOPOS parameter.
2. If none of the list entries have a matching key, and MOVETOKEY is neither the greatest nor least among the list entry keys, then the target position is according to the appropriate key sequence for the list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the key.

,MOVETOLIST=*movetolist*
Use this input parameter to specify the number of the target list where the existing entry being moved or the new entry being created will be placed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the target list.

,MOVETOPOS=HEAD
,MOVETOPOS=TAIL
Use this input parameter to specify the position where the existing or new entry will be placed on the MOVETOLIST.

HEAD

- The entry is placed at the head of the list when:
  - Keyed entries are not supported.
  - Keyed entries are supported, a new entry is being created, and neither MOVETOKEY nor ENTRYKEY are specified.
- The entry is placed at the head of the sublist of identically keyed entries when keyed entries are supported and one or more entries with the same key already exists on the MOVETOLIST list.

TAIL

- The entry is placed at the tail of the list when:
  - Keyed entries are not supported.
  - Keyed entries are supported, a new entry is being created, and neither MOVETOKEY nor ENTRYKEY are specified.
- The entry is placed at the tail of the sublist of identically keyed entries when keyed entries are supported and one or more entries with the same key already exists on the MOVETOLIST list.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*
Use this input parameter to specify a list authority value for the list specified by LISTNUM. If NEWAUTH is omitted, the list authority for the designated list is unchanged.

When the structure is allocated, the authority value for each list is initialized to binary zeros.

Note: The NEWAUTH parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher, except on WRITE_LCONTROLS requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,NEWVERS=*newvers*
Use this input parameter with VERSUPDATE=SET to specify a version number to either replace the version number of the existing entry, or initialize the version number of a new entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the entry version number.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual

storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with the PLISTVER macro.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,UPDATECURSOR=NO
,UPDATECURSOR=YES
Use this input parameter to specify whether the MOVETOLIST list cursor is to be updated to point to another entry on the list.

NO
The list cursor is not updated.

YES
The list cursor is updated to point to the entry that is adjacent to the new or moved entry.

- If LISTDIR=TOHEAD, the list cursor will point to the adjacent entry that is closest to the head of the list.
- If LISTDIR=TOTAIL, the list cursor will point to the adjacent entry that is closest to the tail of the list.

The list cursor is set to binary zeros if its new position would be before the first entry or after the last.

Note: UPDATECURSOR may not be specified when DATAOPER=WRITE and ENTRYTYPE=ANY are specified

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*
Use this input parameter to specify a version number to be compared to the version number of the existing entry. If this request creates a new entry, the VERSCOMP specification is ignored.

The existing entry is moved only if its version number meets the condition specified by the VERSCOMPTYPE parameter. If the entry does not have the specified version number, the request is terminated with no change to the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

,VERSUPDATE=NONE
,VERSUPDATE=INC
,VERSUPDATE=DEC
,VERSUPDATE=SET
Use this input parameter to specify how the entry version number of the moved entry will be updated, or for those cases where a new entry is created, initialized.

VERSUPDATE=NONE
The existing entry's version number is not updated. The new entry's version number is set to binary zeros.

VERSUPDATE=INC
The existing entry's version number is increased by one. The new entry's version number is set to binary zeros, except for the low-order bit, which is set to one.

VERSUPDATE=DEC
The existing entry's version number is decreased by one. The new entry's version number is set to all binary ones.

VERSUPDATE=SET
Both the existing and the new entry's version number is set to the value specified by NEWVERS.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING

8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved, and the designated entry was moved. The following fields in the answer area have been filled in: • LAATOTALCNT - The total count of allocated entries in the list structure. • LAATOTALELECNT - The total count of allocated elements in the list structure. • LAALISTCNT - The count of entries or elements residing on the target list. • LAALCTL (if the request completed prematurely as well) - The list entry controls for the first UNPROCESSED entry in the list sequence. • LAAFIRSTKEY - Indicates a successful MOVE request and the list entry is the only entry on the list with the entry key value. This field is returned only for a list structure allocated in a coupling facility with CFLEVEL=1. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=SET or LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller’s PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller’s PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0837	**Equate Symbol:** IXLRSNCODEBADWRITEADJDATA **Meaning:** Program error. The request specified that adjunct data was to be written, but the area specified by ADJAREA is not addressable. There was no change to the structure. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx083E	**Equate Symbol:** IXLRSNCODEMAXLISTKEY **Meaning:** Program error. The list key to be assigned to an entry that was being created or moved was not valid. Either the list key or the list key plus the list key increment value was greater than the maximum list key. **Action:** Ensure that you specified a correct list key increment. Depending on the protocol you are using for assigning list key values, you might want to issue a WRITE_LCONTROLS request to update either the list key value to a lower value or the maximum list key value to a higher value.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The designated entry has a version number that failed the comparison specified by VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the value specified in VERSCOMP and look at the version returned in the answer area.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by ENTRYID or ENTRYNAME exists in the structure, but does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0841	**Equate Symbol:** IXLRSNCODEBADENTRYNAME **Meaning:** Program error. Entry creation was attempted, but no entry was created for one of the following reasons: • No entry name was specified (and the structure supports names). • The specified entry name is not unique within the structure. The list entry controls for the first entry on the list are returned in the answer area (field LAALCTL). **Action:** Be sure to provide a unique entry name when creating entries to be written to structures that support entry names.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND, or LOCKOPER=NOTHELD with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx084E	**Equate Symbol:** IXLRSNCODEBADMOVETOLIST **Meaning:** Program error. The list number specified for MOVETOLIST exceeds the number of lists defined for the structure. **Action:** Correct MOVETOLIST to specify a list that exists in the structure. The maximum number of lists in a structure is determined by the LISTHEADERS parameter on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data for the first entry to be read in the list. No data is returned. **Action:** If more space is available, specify a larger buffer size, or more buffers, and reissue the request. Check the information returned in the answer area (mapped by IXLYLAA) in the field LAALCTL (mapped by IXLYLCTL).
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. For a request that writes data, the data cannot be fetched from the specified buffer area. For a request that reads data, the data cannot be stored into the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The value specified for ELEMNUM is not valid. **Action:** Correct the ELEMNUM specification to be within the allowable range: from zero to whatever MAXELEMNUM value was returned to the connector in the connect answer area.

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The request attempted to create a new entry or overwrite an existing entry, but the structure is full and cannot accommodate any more entries. **Action:** Determine why the structure is full. You should be monitoring the usage of the structure every time a read or write is done (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure). This data should be evaluated periodically to ensure structure resources are being used efficiently. You might be able to delete existing entries to free up space, rebuild the structure to make it larger if rebuild is allowed (IXLCONN macro, ALLOWREBLD parameter), or alter the size of the structure (IXLALTER macro).

Table 58. Return and Reason Codes for IXLLIST REQUEST=MOVE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C18	**Equate Symbol:** IXLRSNCODELISTFULL **Meaning:** Environmental error. The entry could not be placed on the designated target list because the list is full. **Action:** Either change the maximum number of entries allowed on the list by specifying a new LISTLIMIT on a WRITE_LCONTROLS request. You should be monitoring the usage of the list structure every time a read or write is done. (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure.) This data should be evaluated periodically to ensure structure resources are being used efficiently.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ

Description

A READ request allows you to read a single list entry from the list structure into the storage areas specified by BUFLIST or BUFFER and/or ADJAREA. To designate an entry to be read, specify one of the following parameters or parameter combinations:

- List number (LISTNUM) with list position (LISTPOS)
- Entry identifier (ENTRYID)
- Entry name (ENTRYNAME)
- Locate by list cursor (LOCBYCURSOR) with a list number (LISTNUM)
- Entry key (ENTRYKEY) with a list number (LISTNUM)

Entry keys, however, are not unique within the structure such that different entries may have the same key (a sublist) on the same list. If there is a sublist of entries with the same key as the ENTRYKEY key (or that satisfies the KEYREQTYPE criteria, if specified), the entry that is designated is the entry at the HEAD or TAIL of the sublist as specified by LISTPOS. (KEYREQTYPE allows you to designate an entry with a key that varies from, but is as close as possible to, the specified key.)

With a coupling facility of CFLEVEL=1 or higher, the following additional functions are supported for a READ request:

- You can specify that a list entry is to be read only after a successful comparison of the list authority value for the target list with a user-specified value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. If the request is successful, you can also update the list authority value to a new value that you specify with NEWAUTH.
- You can specify that a list entry be read only if it passes a version number comparison. You specify the version comparison requirements using VERSCOMP and VERSCOMPTYPE.
- There are additional list cursor options. You can update the list cursor conditionally, and you can set the list cursor to point to the current entry instead of the previous or next entry.

A READ request reads any combination of the following types of data for the designated entry:

- List entry controls
- Entry data
- Adjunct data

Whether a particular type of data is returned depends on whether you specified the local storage area to contain that data. Listed below are the types of data that are returned when the indicated storage areas are specified:

- List entry controls, the number of list entries or elements residing on the list, and the total number of allocated entries in the structure are returned in the answer area (ANSAREA) mapped by IXLYLAA.
- Entry data is returned in a buffer (BUFFER) or list of buffers (BUFLIST).
- Adjunct data is returned to the adjunct area (ADJAREA).
- If you do not specify either BUFFER or BUFLIST, or ADJAREA, list entry controls are returned in the answer area.

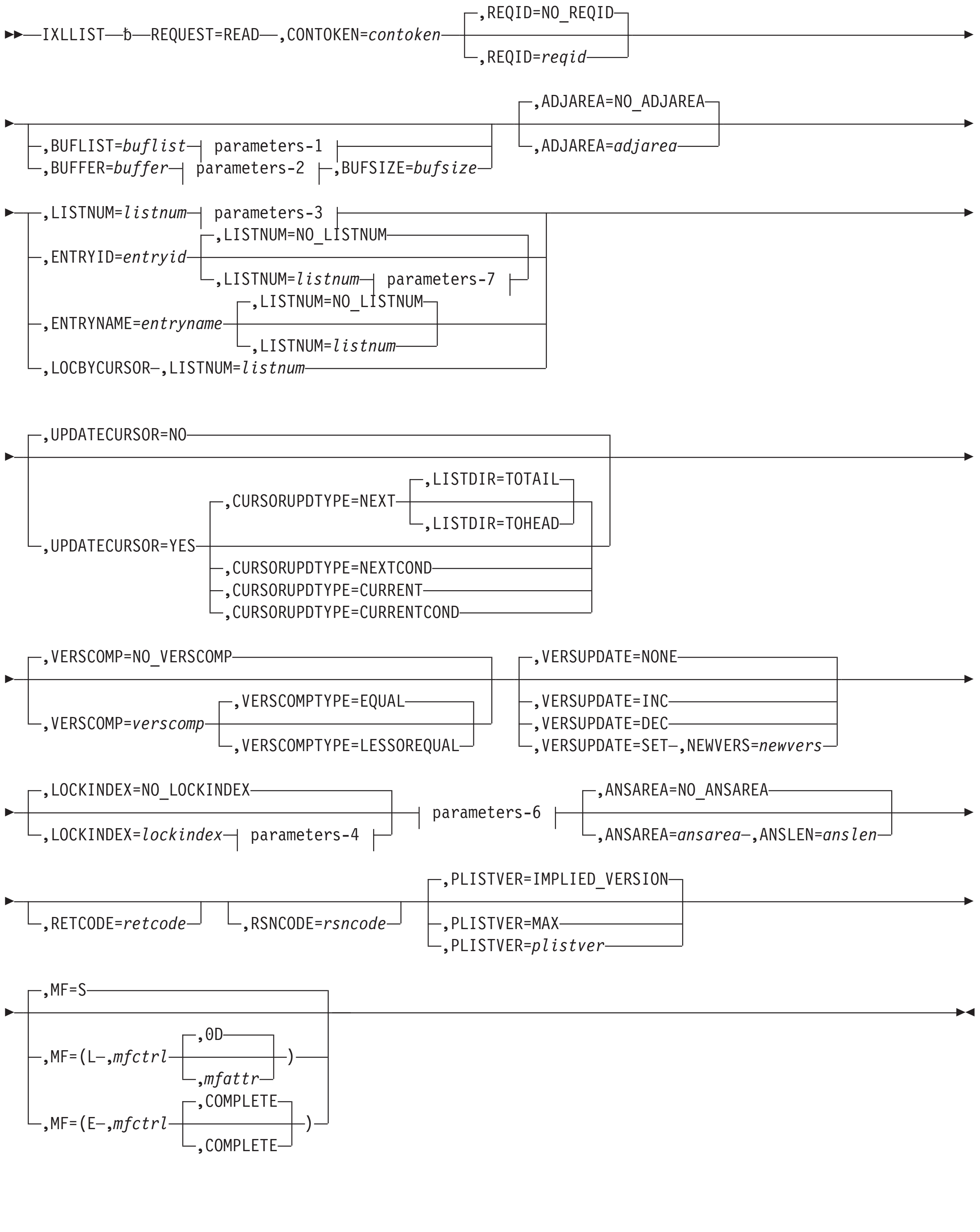
Additionally, you can perform locking functions with a READ request by specifying LOCKOPER. The lock designated by LOCKINDEX will be operated on as specified by LOCKOPER.

Syntax Diagram

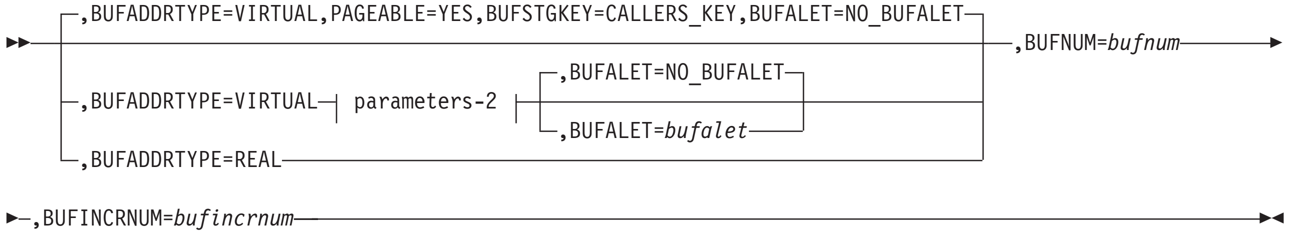
The syntax diagram for IXLLIST REQUEST=READ is as follows:

IXLLIST Macro

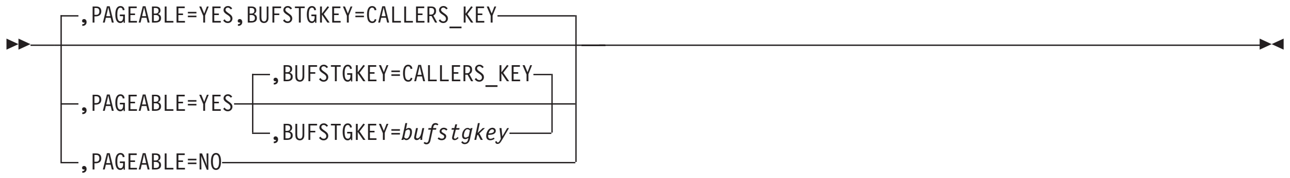
main diagram



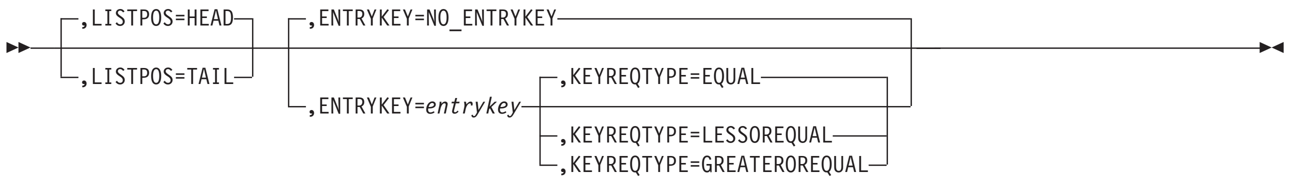
parameters-1



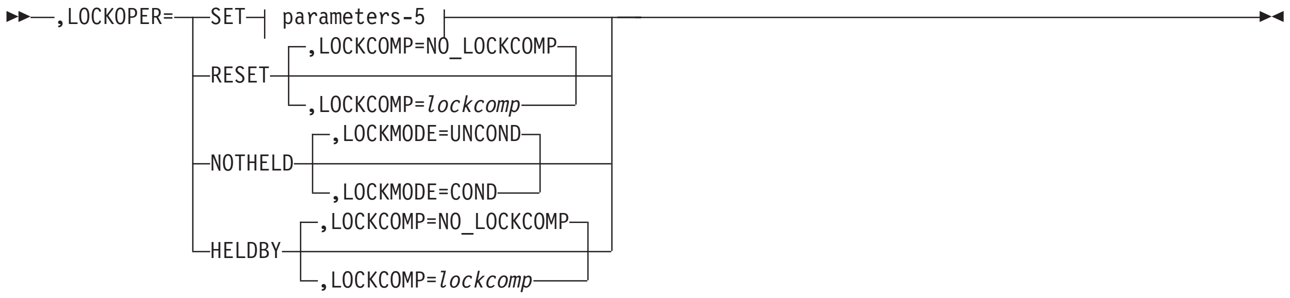
parameters-2



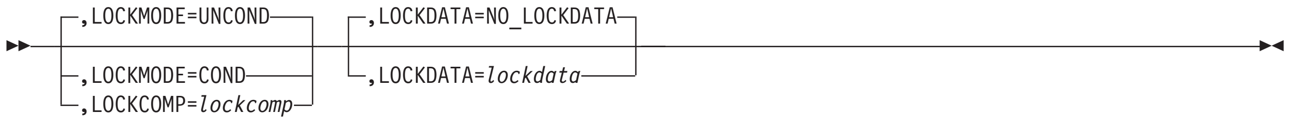
parameters-3



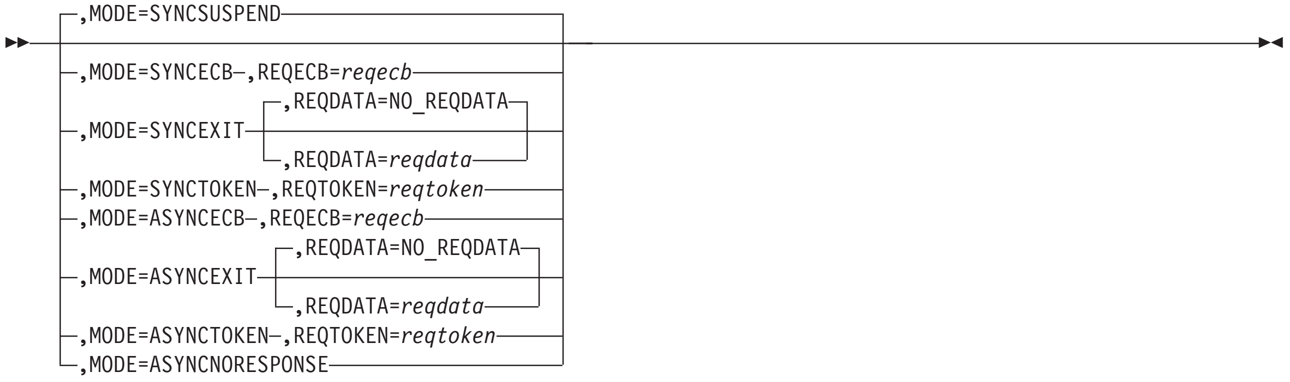
parameters-4



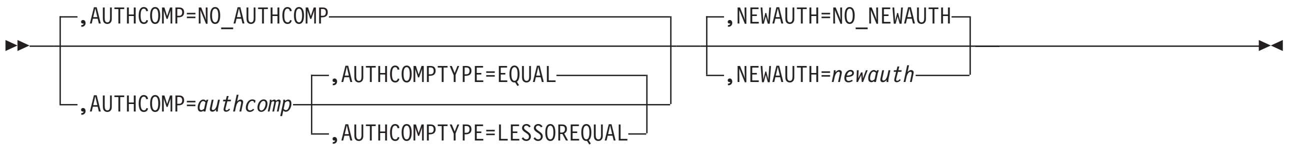
parameters-5



parameters-6



parameters-7



Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ are listed in alphabetical order. Default values are underlined:

REQUEST=READ
Use this input parameter to read a single list entry.

,ADJAREA=<u>NO_ADJAREA</u>
,ADJAREA=*adjarea*
Use this output parameter to specify a storage area to contain the adjunct data that was read from the designated entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area where the adjunct data will be put.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion of the request, the following information is returned to the answer area:
- List entry controls of the designated entry (field LAALCTL).
- The number of list entries or elements residing on the list (field LAALISTCNT).
- The total number of allocated entries in the structure (field LAATOTALCNT).
- The total number of allocated elements in the structure (field LAATOTALELECNT).
- If none of BUFLIST, BUFFER, or ADJAREA is specified, only list entry controls are returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned when the request completes.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold the entry data that is read from the designated entry.

You can define the buffer size to be a total of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

See the BUFSIZE parameter description for defining the size of the buffer.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that will contain the entry data when the request completes.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold the entry data that is read from the designated entry. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:

- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

  Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single buffer, even if the buffers are not contiguous.

  See the BUFNUM and BUFINCRNUM keyword descriptions for specifying the number and size of buffers.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CURSORUPDTYPE=NEXT
,CURSORUPDTYPE=NEXTCOND
,CURSORUPDTYPE=CURRENT
,CURSORUPDTYPE=CURRENTCOND
Use this input parameter to specify how the list cursor is to be updated when UPDATECURSOR=YES is specified.

Note: The NEXTCOND, CURRENT, and CURRENTCOND values are valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

CURSORUPDTYPE=NEXT
Update the list cursor to point to the list entry before or after the target entry.

- For MOVE requests that specify DATAOPER=WRITE and ENTRYTYPE=ANY and that result in the creation of a new entry, the cursor for the list specified by MOVETOLIST is updated. The direction of the cursor update depends on the value specified for MOVETOPOS.
- For all other requests, the cursor for the source list is updated. The direction of the cursor update depends on the value specified for LISTPOS or LISTDIR.

CURSORUPDTYPE=NEXTCOND
Update the list cursor to point to the list entry before or after the target entry only if the cursor points to the target list entry, and that list entry is moved or deleted.

Set the list cursor direction with SETCURSOR on a WRITE_LCONTROLS request.

The list cursor will be reset to binary zeros if either:

- The entry is the last entry on the list and the list cursor direction is set to a head-to-tail direction.
- The entry is the first entry on the list and the list cursor is set to a tail-to-head direction.

CURSORUPDTYPE=CURRENT
Set the list cursor to point to the list entry processed for the request.

If the list entry is deleted or moved to another list, then the list cursor will be reset to binary zeros.

CURSORUPDTYPE=CURRENTCOND
Set the list cursor to point to the list entry processed only if the list cursor value currently is zero and the target list entry is not deleted or moved to another list.

If the list entry is deleted or moved to another list, then the list cursor will remain binary zeros.

,ENTRYID=*entryid*
Use this input parameter to designate the entry identifier of the entry to be read.

Each entry identifier, which is assigned by list services when the entry is created, is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=NO_ENTRYKEY
,ENTRYKEY=*entrykey*
Use this input parameter with LISTNUM to designate the entry key of the entry to be read.

If there is a sublist of entries on the list all with the same key as the ENTRYKEY key, the LISTPOS parameter determines whether entry at the HEAD or the TAIL of the sublist is designated.

Specify ENTRYKEY only for structures that support keyed entries.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*
Use this input parameter to designate the name of the entry to be read.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL
Use this input parameter to specify how, if at all, the entry key of the entry to be read can differ from the entry key specified by ENTRYKEY. KEYREQTYPE applies only to locating an existing entry; it does not determine the key of a new entry.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

Notes:

1. When no entries on the list meet the requirements of the KEYREQTYPE, the request will be failed with a return code X'8' and reason code IXLRSNCODEOENTRY.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, the entry with an entry key value closest to the value specified will be

selected. When multiple entries have the same entry key value, LISTPOS is used to resolve whether the first or last entry with the entry key value is selected.

,LISTDIR=TOTAIL
,LISTDIR=TOHEAD

Use this input parameter with UPDATECURSOR to specify the direction in which the list cursor is moved as a result of this request. See the UPDATECURSOR parameter for a full description of LISTDIR.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entry to be read resides. Use LISTNUM in the following ways:

- With LISTPOS and/or ENTRYKEY to designate the entry to be read
- With either ENTRYID or ENTRYNAME to verify that the designated entry resides on the LISTNUM list before proceeding with the request
- With LOCBYCURSOR to designate the entry to be read

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTPOS=HEAD
,LISTPOS=TAIL

Use this input parameter with LISTNUM to designate the entry to be read. The designated entry is at the HEAD or the TAIL of a list or sublist:

- If you specify ENTRYKEY and the list contains a sublist of one or more entries with the same key (or that satisfies the KEYREQTYPE criteria), then:
  - LISTPOS=HEAD designates the entry at the head of the sublist.
  - LISTPOS=TAIL designates the entry at the tail of the sublist.
- If you do not specify ENTRYKEY, then:
  - LISTPOS=HEAD designates the entry at the head of the list.
  - LISTPOS=TAIL designates the entry at the tail of the list.

,LOCBYCURSOR

Use this input parameter with LISTNUM to designate the entry to be read. The designated entry is the entry to which the LISTNUM list cursor is pointing.

Be aware that the list cursor could have been reset to zeros by a previous request that, for instance, deleted or moved the entry to which the list cursor points, or updated the list cursor after the last entry on the list had been processed. See *z/OS MVS Programming: Sysplex Services Guide* for more information about using the list cursor.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA
,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:

- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND

Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND

If the specified lock is held by another user, your request is either:

- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND

The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET

Set the lock.

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released or fails, depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

RESET

Reset the lock.

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

NOTHELD
The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released or your request fails, depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

HELDBY
Processing is determined by which connector is holding the lock.

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT

,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*

Use this input parameter to specify a list authority value for the list specified by LISTNUM. If NEWAUTH is omitted, the list authority for the designated list is unchanged.

When the structure is allocated, the authority value for each list is initialized to binary zeros.

Note: The NEWAUTH parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher, except on WRITE_LCONTROLS requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,NEWVERS=*newvers*

Use this input parameter with VERSUPDATE=SET to specify a new version number to replace the old version number of the entry to be read.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the entry version number.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,UPDATECURSOR=NO
,UPDATECURSOR=YES

Use this input parameter to specify whether the LISTNUM list cursor is to be updated to point to another entry on the list.

NO

The list cursor is not updated.

YES

The list cursor is updated to point to the entry that is adjacent to the entry that is read:

- If LISTDIR=TOHEAD, the list cursor will point to the adjacent entry that is closest to the head of the list.

- If LISTDIR=TOTAIL, the list cursor will point to the adjacent entry that is closest to the tail of the list.

The list cursor is set to binary zeros if its new position would be before the first entry or after the last.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of the designated entry to be read. The entry is read only if its version number meets the condition specified by the VERSCOMPTYPE parameter. If the designated entry does not have the specified version number, the request is terminated with no change to the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

,VERSUPDATE=NONE
,VERSUPDATE=INC
,VERSUPDATE=DEC
,VERSUPDATE=SET

Use this input parameter to specify how the version number of the designated entry will be updated.

VERSUPDATE=NONE
The version number is not updated.

VERSUPDATE=INC
The version number is increased by one.

VERSUPDATE=DEC
The version number is decreased by one.

VERSUPDATE=SET
The version number is set to the value specified by NEWVERS.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved and the following fields in the answer area have been filled in: • LAATOTALCNT - The total count of allocated entries in the list structure. • LAATOTALELECNT - The total count of allocated elements in the list structure. • LAALISTCNT - The count of entries or elements residing on the processed list. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=SET or LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller’s PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The designated entry has a version number that failed the comparison specified by VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the value specified in VERSCOMP and look at the version returned in the answer area.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by ENTRYID or ENTRYNAME exists in the structure, but does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND, or LOCKOPER=NOTHELD with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data for the first entry to be read in the list. No data is returned. **Action:** If more space is available, specify a larger buffer size, or more buffers, and reissue the request. Check the information returned in the answer area (mapped by IXLYLAA) in the field LAALCTL (mapped by IXLYLCTL).
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 59. Return and Reason Codes for IXLLIST REQUEST=READ Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ_EMCONTROLS

Description

A READ_EMCONTROLS request allows you to retrieve control information about your registered interest in a designated sublist within the list structure that you are monitoring. The sublist is identified by list number (LISTNUM) and entry key (ENTRYKEY). The information is returned in the answer area (ANSAREA).

You can issue a READ_EMCONTROLS request only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. READ_EMCONTROLS requests issued for a list structure allocated in a coupling facility of CFLEVEL=0, 1, or 2 will fail.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=READ_EMCONTROLS is as follows:

main diagram

```
                                                                ┌─,REQID=NO_REQID─┐
►►──IXLLIST──ƀ──REQUEST=READ_EMCONTROLS──,CONTOKEN=contoken─────┼─────────────────┼─,LISTNUM=listnum──────────►
                                                                └─,REQID=reqid────┘

                                           ┌─,ANSAREA=NO_ANSAREA──────────────┐
►─,ENTRYKEY=entrykey──┤ parameters-1 ├─────┼──────────────────────────────────┼──┬────────────────────┬────────►
                                           └─,ANSAREA=ansarea─,ANSLEN=anslen──┘  └─,RETCODE=retcode───┘

                                ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S────────────────────────────────────┐
►──┬────────────────────┬───────┼───────────────────────────┼──┼──────────────────────────────────────────┼──────►◄
   └─,RSNCODE=rsncode───┘       ├─,PLISTVER=MAX─────────────┤  │                         ┌─,0D──────┐     │
                                └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl──────────┼──────────┼─)───┤
                                                               │                         └─,mfattr──┘     │
                                                               │                         ┌─,COMPLETE─┐    │
                                                               └─,MF=(E─,mfctrl──────────┼───────────┼─)──┘
                                                                                         └─,COMPLETE─┘
```

parameters-1

```
   ┌─,MODE=SYNCSUSPEND───────────────────────────────┐
►►─┼─────────────────────────────────────────────────┼──────────────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb────────────────────┤
   │                  ┌─,REQDATA=NO_REQDATA─┐        │
   ├─,MODE=SYNCEXIT───┼─────────────────────┼────────┤
   │                  └─,REQDATA=reqdata────┘        │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken──────────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb───────────────────┤
   │                   ┌─,REQDATA=NO_REQDATA─┐       │
   ├─,MODE=ASYNCEXIT───┼─────────────────────┼───────┤
   │                   └─,REQDATA=reqdata────┘       │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
   └─,MODE=ASYNCNORESPONSE───────────────────────────┘
```

Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_EMCONTROLS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_EMCONTROLS
Use this input parameter to specify that event monitor control information that represents your registered monitoring interest in the sublist specified by LISTNUM and ENTRYKEY be returned in the answer area.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request for event monitor control information, the answer area contains:

- The connection identifier of the connector associated with the EMC (field LAAREMC_CONID)
- An indicator as to whether the EMC is queued to the event queue of the connector identified by LAAREMC_CONID (field LAAREMC_EMCQUEUED)
- The list number of the list with which the EMC is associated (field LAAREMC_LISTNUM)
- The list entry key of the sublist with which the EMC is associated (field LAAREMC_ENTRYKEY)
- The user notification control data that was supplied by the connector when this EMC was established to monitor the indicated sublist (field LAAREMC_UNC).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,ENTRYKEY
Use this input parameter to specify the entry key to be used in conjunction with LISTNUM to designate the sublist for which control information is to be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key of the sublist.

,LISTNUM=*listnum*
Use this input parameter to specify the number of the list containing the sublist for which control information is to be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,MF=<u>S</u>
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**

,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID

,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*

Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*

Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 60. Return and Reason Codes for IXLLIST REQUEST=READ_EMCONTROLS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.

Table 60. Return and Reason Codes for IXLLIST REQUEST=READ_EMCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.

Table 60. Return and Reason Codes for IXLLIST REQUEST=READ_EMCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated event monitor controls object (EMC) does not exist for the user of the designated sublist. **Action:** None necessary. However, if this return code and reason code are unexpected, you should examine the parameters specified for the list number and entry key on the invocation of this macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.

Table 60. Return and Reason Codes for IXLLIST REQUEST=READ_EMCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.

Table 60. Return and Reason Codes for IXLLIST REQUEST=READ_EMCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ_EQCONTROLS

Description

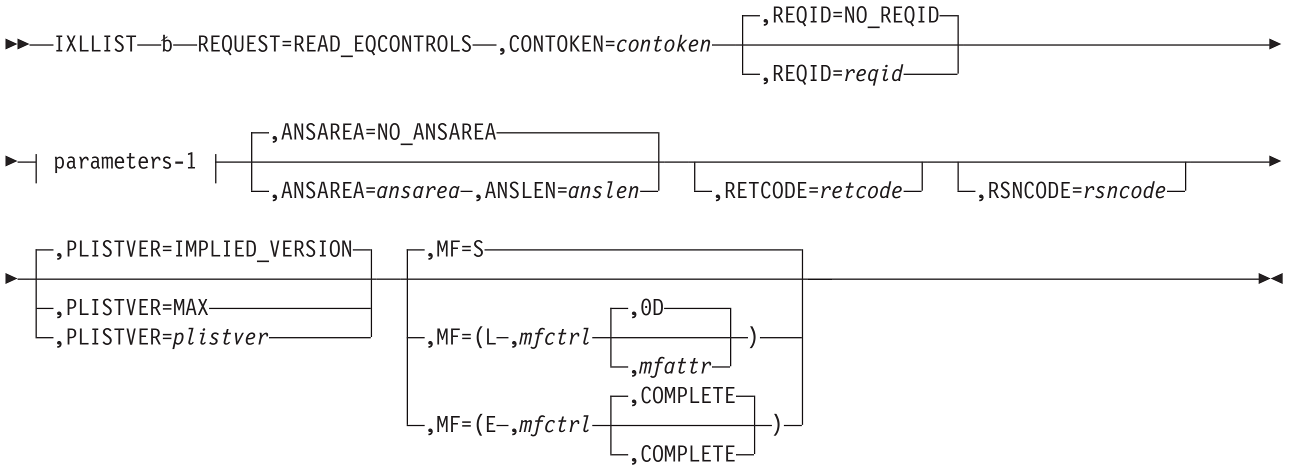
A READ_EQCONTROLS request allows you to retrieve control information about your event queue. The information is returned in the answer area (ANSAREA).

You can issue a READ_EQCONTROLS request only for keyed list structures allocated in a coupling facility of CFLEVEL=3 or higher. READ_EQCONTROLS requests issued for a list structure allocated in a coupling facility of CFLEVEL=0, 1, or 2 will fail.

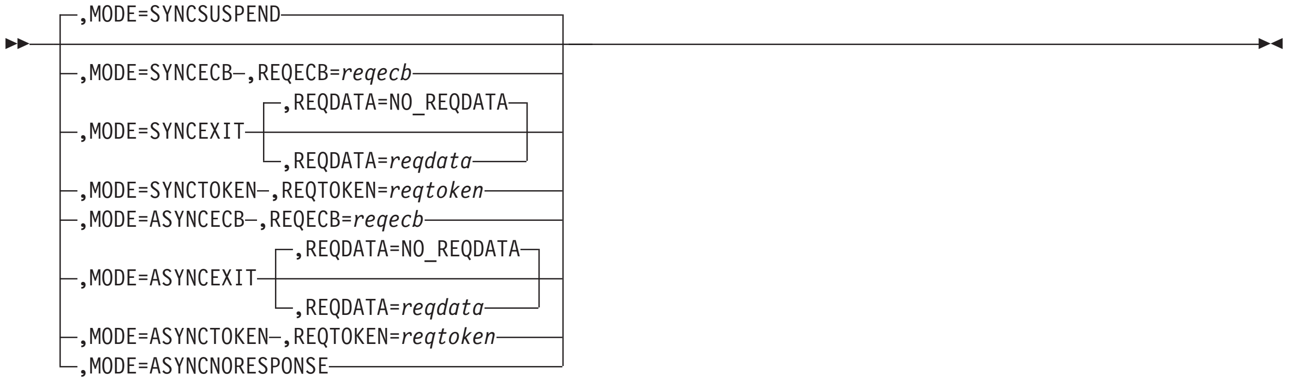
Syntax Diagram

The syntax diagram for IXLLIST REQUEST=READ_EQCONTROLS is as follows:

main diagram



parameters-1



Note: If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_EQCONTROLS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_EQCONTROLS
Use this input parameter to specify that event queue control information associated with your event queue is to be retrieved.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA. Upon successful completion of a request for event queue control information, the answer area contains:

- An indicator as to whether the system is to drive the connector's list transition exit when this event queue changes from empty to nonempty (field LAAREQC_DRIVEEXIT)
- An indicator as to whether the user is currently monitoring the event queue (field LAAREQC_MONITORINGACTIVE)
- The vector index associated with the monitored event queue (field LAAREQC_VECTORINDEX)
- The number of event monitor controls (EMCs) queued to the event queue (field LAAREQC_EMCQUEUEDCNT)
- The approximate number of empty to nonempty event queue transitions that have occurred (field LAAREQC_EVENTTRAN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) to contain the information.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=<u>SYNCSUSPEND</u>
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN

The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB

The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT

The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.

Table 61. Return and Reason Codes for IXLLIST REEQUEST=READ_EQCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ_LCONTROLS

Description

A READ_LCONTROLS request allows you to retrieve list control and monitoring for a specified list (LISTNUM). The information is returned in the answer area (ANSAREA) and either a buffer (BUFFER) or list of buffers (BUFLIST). See the ANSAREA, BUFFER, or BUFLIST parameter descriptions for the specific information returned by this request.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=READ_LCONTROLS is as follows:

main diagram

```
                                                            ┌─,REQID=NO_REQID─┐
►►──IXLLIST──␢──REQUEST=READ_LCONTROLS──,CONTOKEN=contoken──┼─────────────────┼────────────────────────────►
                                                            └─,REQID=reqid────┘

►──┬─,BUFLIST=buflist─┤ parameters-1 ├──┬──,LISTNUM=listnum──────────────────────────────────────────────►
   └─,BUFFER=buffer─┤ parameters-2 ├────┘

   ┌─,MODE=SYNCSUSPEND──────────────────────────────┐   ┌─,ANSAREA=NO_ANSAREA──────────────┐
►──┼────────────────────────────────────────────────┼───┼──────────────────────────────────┼──────────►
   ├─,MODE=SYNCECB─,REQECB=reqecb───────────────────┤   └─,ANSAREA=ansarea─,ANSLEN=anslen──┘
   │                  ┌─,REQDATA=NO_REQDATA─┐       │
   ├─,MODE=SYNCEXIT───┼─────────────────────┼───────┤
   │                  └─,REQDATA=reqdata────┘       │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─────────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb──────────────────┤
   │                   ┌─,REQDATA=NO_REQDATA─┐      │
   ├─,MODE=ASYNCEXIT───┼─────────────────────┼──────┤
   │                   └─,REQDATA=reqdata────┘      │
   └─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken────────────┘

                                                  ┌─.PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬───┬───────────────────┬──┼───────────────────────────┼─────────────────────────►
   └─,RETCODE=retcode──┘   └─,RSNCODE=rsncode──┘  ├─.PLISTVER=MAX─────────────┤
                                                  └─.PLISTVER=plistver────────┘

   ┌─,MF=S───────────────────────────────┐
►──┼─────────────────────────────────────┼──────────────────────────────────────────────────────────────►◄
   │                   ┌─,0D──────┐      │
   ├─,MF=(L─,mfctrl────┼──────────┼─)────┤
   │                   └─,mfattr──┘      │
   │                   ┌─,COMPLETE─┐     │
   └─,MF=(E─,mfctrl────┼───────────┼─)───┘
                       └─,COMPLETE─┘
```

parameters-1

```
     ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─┐
►►───┼───────────────────────────────────────────────────────────────────────────┼───────────────►◄
     │                                                ┌─,BUFALET=NO_BUFALET─┐    │
     ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├──────────┼─────────────────────┼────┤
     │                                                └─,BUFALET=bufalet────┘    │
     └─,BUFADDRTYPE=REAL─────────────────────────────────────────────────────────┘
```

parameters-2

```
     ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────┐
►►───┼───────────────────────────────────────────┼──────────────────────────────────────────────────►◄
     │                  ┌─,BUFSTGKEY=CALLERS_KEY─┐│
     ├─,PAGEABLE=YES────┼────────────────────────┼┤
     │                  └─,BUFSTGKEY=bufstgkey───┘│
     └─,PAGEABLE=NO───────────────────────────────┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, BUFLIST and BUFFER may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_LCONTROLS are listed in alphabetical order. Default values are underlined:

REQUEST=READ_LCONTROLS
Use this input parameter to specify that control information for the list specified by LISTNUM be returned.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain list control information about the LISTNUM list that is returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

When the request successfully completes, the following information is returned to the answer area:

- The total number of entries or elements currently on the list (field LAALISTCNT).
- The maximum number of entries or elements allowed to reside on the list (field LAALISTLIMIT).
- The approximate number of list transitions from the empty to the non-empty state (field LAALISTTRAN).
- The user-defined list description (field LAALISTDESC) and list authority (field LAALISTAUTH).
- The number of list monitoring information elements returned in the BUFFER area or BUFLIST buffers (field LAALMICNT).
- The list entry identifier of the entry to which the list cursor points (field LAALISTCURSOR).
- The current value of the list cursor direction indicator (field LAACURSORDIR). Returned only for a structure allocated in a CFLEVEL=1 or higher coupling facility.
- The list controls key value (field LAALISTKEY). Returned only for a structure allocated in a CFLEVEL=1 or higher coupling facility.
- The list controls maximum list key value (field LAAMAXLISTKEY). Returned only for a structure allocated in a CFLEVEL=1 or higher coupling facility.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information returned from the request will be put.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,BUFADDRTYPE=<u>VIRTUAL</u>

,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this output parameter to specify a buffer area to hold the returned list monitoring data for the LISTNUM list.

You must define the buffer to be 4096 bytes on a 4096-byte boundary.

Once the request completes, the buffer contains, starting at offset zero, an array of list monitoring information. The relative position of an element in the array associates that element with a connection identifier. The first array element is associated with a connection identifier of zero, and is reserved. The format of each array element is described by mapping macro IXLYLMI. The array elements are numbered from 0 to LAALMICNT-1.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4096-byte area to contain the data returned by the request.

,BUFLIST=*buflist*
Use this input parameter to specify a buffer to hold the returned list monitoring information for the LISTNUM list.

BUFLIST specifies a 128-byte storage area that contains the address of a single buffer. (Unlike any other IXLLIST request, for a READ_LCONTROLS request, only one buffer can be passed.) The address of the buffer should be placed in the second four bytes of the 128-byte area, beginning at offset zero. The rest of the area is ignored.

You must define the buffer to be 4096 bytes on a 4096-byte boundary.

Once the request completes, the buffer contains, starting at offset zero, an array of list monitoring information. The relative position of an element in the array associates that element with a connection identifier. The first element is associated with a connection identifier of zero, and is reserved. The format of each array element is described by mapping macro IXLYLMI.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the address of the buffer.

,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffer specified by BUFLIST or BUFFER.

If you do not specify BUFSTGKEY, all references to the buffer are assumed to be in your PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the storage key in the format kkkkxxxx in which only the left four bits are used. (The right four bits are ignored.)

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,LISTNUM=*listnum*

Use this input parameter to specify the number of the list for which list control and monitoring information will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion if the request is processed asynchronously.

See *z/OS MVS Programming: Sysplex Services Guide* for more information on understanding synchronous and asynchronous cache operations.

SYNCSUSPEND
The request processes synchronously. If necessary, the request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned to the area specified by REQTOKEN. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual

storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

ABEND Codes

Abend X'026' (see *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- GPR 0 (and RSNCODE, if specified) contains a reason code, if applicable..

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or the buffer in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area to pageable storage or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or the buffer in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or the buffer in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. A real storage address was provided in the BUFLIST list, but the buffer is not addressable in real storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • The requirements for BUFLIST or BUFFER. • Buffer pointer(s) in the BUFLIST. • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 62. Return and Reason Codes for IXLLIST REQUEST=READ_LCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ_LIST

Description

A READ_LIST request allows you to read multiple entries from a single list into the storage areas specified by ANSAREA and/or BUFLIST or BUFFER and/or ADJAREA.

Entries are read in sequence from a starting list entry to the head or tail of the list on which the starting entry resides. The direction of processing (toward the head or toward the tail) depends on the direction you specify on the LISTDIR parameter. The starting entry must exist or the request fails.

For each processed entry, any combination of the following types of data can be read depending on what you specify on the TYPE parameter:

- List entry controls
- Entry data
- Adjunct data

Additionally, you can perform locking functions with a READ_LIST request by specifying LOCKOPER. The lock designated by LOCKINDEX will be operated on as specified by LOCKOPER.

With a coupling facility of CFLEVEL=1 or higher, the following additional functions are supported for a READ_LIST request:

- You can specify that each list entry is to be read only after a successful comparison of the list authority value for the target list with a user-specified value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. If the request is successful, you also can update the list authority value to a new value that you specify with NEWAUTH.
- You can specify that a list entry is to be read only after a successful comparison of the entry key value with a user-specified value (KEYCOMP). If the comparison is not successful, the current list entry is not read and processing continues with the next entry to be considered.
- Version number comparison also is enhanced to allow for an additional less-than-or-equal comparison operation.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=READ_LIST is as follows:

IXLLIST Macro

main diagram

```
►►──IXLLIST──b──REQUEST=READ_LIST──,TYPE=──┬─ENTDATA──────────────────┬──,CONTOKEN=contoken──────────►
                                           ├─ADJDATA─,ADJAREA=adjarea─┤
                                           └─ECONTROLS────────────────┘

   ┌─,REQID=NO_REQID─┐
►──┼─────────────────┼──┬─,BUFLIST=buflist─┤ parameters-1 ├─────────────────────┬──────────────►
   └─,REQID=reqid────┘  └─,BUFFER=buffer─┤ parameters-2 ├─,BUFSIZE=bufsize─┘

                                                        ┌─,LISTDIR=TOTAIL─┐  ┌─,KEYCOMP=NO_KEYCOMP─┐
►──┬─,LISTNUM=listnum─┤ parameters-3 ├──────────────┬──┼─────────────────┼──┼─────────────────────┼──►
   │                      ┌─,LISTNUM=NO_LISTNUM─┐   │  └─,LISTDIR=TOHEAD─┘  └─,KEYCOMP=keycomp────┘
   ├─,ENTRYID=entryid─────┼─────────────────────┼───┤
   │                      └─,LISTNUM=listnum────┘   │
   │                          ┌─,LISTNUM=NO_LISTNUM─┐│
   ├─,ENTRYNAME=entryname─────┼─────────────────────┼┤
   │                          └─,LISTNUM=listnum────┘│
   └─,LOCBYCURSOR─,LISTNUM=listnum──────────────────┘

   ┌─,VERSCOMP=NO_VERSCOMP─────────────────────────┐  ┌─,LOCKINDEX=NO_LOCKINDEX──────────────────┐
►──┼───────────────────────────────────────────────┼──┼──────────────────────────────────────────┼──►
   │                      ┌─,VERSCOMPTYPE=EQUAL───────┐│  └─,LOCKINDEX=lockindex─┤ parameters-4 ├──┘
   └─,VERSCOMP=verscomp───┼───────────────────────────┼┘
                          └─,VERSCOMPTYPE=LESSOREQUAL─┘

                      ┌─,ANSAREA=NO_ANSAREA──────────────┐
►──┤ parameters-5 ├──┼──────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬──►
                      └─,ANSAREA=ansarea─,ANSLEN=anslen─┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────┼──┼───────────────────────────────────────┼──────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D──────┐         │
   └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl──┼──────────┼─)──────┤
                                  │                  └─,mfattr──┘         │
                                  │                  ┌─,COMPLETE─┐        │
                                  └─,MF=(E─,mfctrl──┼───────────┼─)─────┘
                                                     └─,COMPLETE─┘
```

parameters-1

```
     ┌─,BUFADDRTYPE=VIRTUAL,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY,BUFALET=NO_BUFALET─┐
►►──┼────────────────────────────────────────────────────────────────────────────┼─,BUFNUM=bufnum──►◄
     │                                          ┌─,BUFALET=NO_BUFALET─┐           │
     ├─,BUFADDRTYPE=VIRTUAL─┤ parameters-2 ├────┼─────────────────────┼───────────┤
     │                                          └─,BUFALET=bufalet────┘           │
     └─,BUFADDRTYPE=REAL──────────────────────────────────────────────────────────┘
```

parameters-2

```
     ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY───────┐
►►──┼───────────────────────────────────────────┼──────────────────────────────────────────►◄
     │                ┌─,BUFSTGKEY=CALLERS_KEY─┐ │
     ├─,PAGEABLE=YES──┼────────────────────────┼─┤
     │                └─,BUFSTGKEY=bufstgkey───┘ │
     └─,PAGEABLE=NO─────────────────────────────┘
```

parameters-3

```
   ┌─,LISTPOS=HEAD─┐   ┌─,ENTRYKEY=NO_ENTRYKEY──────────────────────────────────┐
►►─┴─,LISTPOS=TAIL─┴───┼────────────────────────────────────────────────────────┼──────────────►
                       │                      ┌─,KEYREQTYPE=EQUAL──────────┐    │
                       └─,ENTRYKEY=entrykey───┼────────────────────────────┼────┘
                                              ├─,KEYREQTYPE=LESSOREQUAL────┤
                                              └─,KEYREQTYPE=GREATEROREQUAL─┘

  ┌─,AUTHCOMP=NO_AUTHCOMP───────────────────────────────────┐
►─┼─────────────────────────────────────────────────────────┼──────────────────────────────►◄
  │                        ┌─,AUTHCOMPTYPE=EQUAL───────┐    │
  └─,AUTHCOMP=authcomp─────┼───────────────────────────┼────┘
                           └─,AUTHCOMPTYPE=LESSOREQUAL─┘
```

parameters-4

```
►►─,LOCKOPER=─┬─NOTHELD─,LOCKMODE=COND──────────────┬──────────────────────────────────────►◄
              │         ┌─,LOCKCOMP=NO_LOCKCOMP─┐   │
              └─HELDBY──┼───────────────────────┼───┘
                        └─,LOCKCOMP=lockcomp────┘
```

parameters-5

```
   ┌─,MODE=SYNCSUSPEND────────────────────────────┐
►►─┼──────────────────────────────────────────────┼────────────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb─────────────────┤
   │                 ┌─,REQDATA=NO_REQDATA─┐      │
   ├─,MODE=SYNCEXIT──┼─────────────────────┼──────┤
   │                 └─,REQDATA=reqdata────┘      │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken───────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb────────────────┤
   │                  ┌─,REQDATA=NO_REQDATA─┐     │
   ├─,MODE=ASYNCEXIT──┼─────────────────────┼─────┤
   │                  └─,REQDATA=reqdata────┘     │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken──────────┤
   └─,MODE=ASYNCNORESPONSE────────────────────────┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, BUFFER, BUFLIST, and LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_LIST are listed in alphabetical order. Default values are underlined:

REQUEST=READ_LIST
  Use this input parameter to read multiple entries from a single list.

,ADJAREA=*adjarea*
  Use this output parameter to specify a storage area to contain the adjunct data that was read from the first processed entry. Adjunct data is returned only if ADJDATA is specified on the TYPE parameter.

  Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area (the adjunct area) to contain the adjunct data.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion of the request, the answer area contains:

- List entry controls (field LAARLRMLCTLS mapped by IXLYLCTL) from the first processed list entry (if ECONTROLS is specified on the TYPE parameter).
- The number of processed entries (field LAAREADCNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information from the request will be put.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this output parameter to specify a buffer area to hold data that is read from list entries.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes
- Is less than or equal to 65536 bytes
- Starts on a 4096-byte boundary

  See the BUFSIZE parameter description for specifying the size of the buffer.

The type of data returned to the buffer depends on which types of data you request on the TYPE parameter. See *z/OS MVS Programming: Sysplex Services Guide* for more detailed information about how data is returned from this request.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) where the data read from list entries will be put.

,BUFLIST=*buflist*
Use this output parameter to specify a list of buffers to hold data that is read from list entries. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:
- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  – The left (high-order) four bytes are reserved and
  – The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.

  Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single buffer, even if the buffers are not contiguous.

  See the BUFNUM parameter description to specify the number of buffers in the buffer list.

The type of data returned to the buffers depends on which types of data you request on the TYPE parameter. See *z/OS MVS Programming: Sysplex Services Guide* for more detailed information about how data is returned from this request.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,ENTRYID=*entryid*
Use this input parameter to designate the entry identifier of the starting entry to be read.

Each entry identifier, which is assigned by list services when the entry is created, is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=NO_ENTRYKEY
,ENTRYKEY=*entrykey*
Use this input parameter with LISTNUM to designate the entry key of the starting entry to be read.

If there is a sublist of entries on the list all with the same key as the ENTRYKEY key, the LISTPOS parameter determines whether the entry at the HEAD or TAIL of the sublist is designated.

Specify ENTRYKEY only for structures that support keyed entries.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*

Use this input parameter to designate the name of the starting entry to be read.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,KEYCOMP=NO_KEYCOMP
,KEYCOMP=*keycomp*

Use this input parameter to specify a value to be compared to the entry key of each list entry in the designated list.

If the list entry that is to be processed does not have an entry key value that is equal to the specified KEYCOMP value, the entry is not processed. Processing continues with the next entry.

Note: The KEYCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the value to be compared to the entry key of the designated list entry.

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL

Use this input parameter to specify how, if at all, the entry key of the entry to be read can differ from the entry key specified by ENTRYKEY. KEYREQTYPE applies only to locating an existing entry; it does not determine the key of a new entry.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

Notes:

1. When no entries on the list meet the requirements of the KEYREQTYPE, the request will be failed with a return code X'8' and reason code IXLRSNCODEOENTRY.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, the entry with an entry key value closest to the value specified will be selected. When multiple entries have the same entry key value, LISTPOS is used to resolve whether the first or last entry with the entry key value is selected.

,LISTDIR=TOHEAD
,LISTDIR=TOTAIL

Use this input parameter to specify the direction of processing from the designated starting entry.

TOHEAD
The direction of processing is from the designated entry to the head of the list.

TOTAIL
The direction of processing is from the designated entry to the tail of the list.

Note: See the LISTPOS parameter description for a note about the valid combinations of LISTDIR and LISTPOS.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entries to be read reside. Use LISTNUM in the following ways:

- With LISTPOS and/or ENTRYKEY to designate the starting entry to be read.
- With either ENTRYID or ENTRYNAME to verify that the designated starting entry resides on the list specified by LISTNUM before proceeding with the request.
- With LOCBYCURSOR to designate the starting entry to be read.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTPOS=HEAD
,LISTPOS=TAIL

Use this input parameter with LISTNUM to designate the starting entry to be read. The designated entry is at the HEAD or the TAIL of a list or sublist:

- If you specify ENTRYKEY and the list contains a sublist of one or more entries with the same key (or that satisfies the KEYREQTYPE criteria, if specified), then:
  - LISTPOS=HEAD designates the entry at the head of the sublist.
  - LISTPOS=TAIL designates the entry at the tail of the sublist.
- If you do not specify ENTRYKEY, then:
  - LISTPOS=HEAD designates the entry at the head of the list.
  - LISTPOS=TAIL designates the entry at the tail of the list.

Note: The direction of processing is always from an entry at one end of a list or sublist, through to the entry at the opposite end of the list. For instance, list services will process from the head of a sublist to the tail of the list, but not from the head of a sublist to the head of the list. As such, only the following combinations of LISTPOS and LISTDIR are valid:

- LISTPOS=HEAD and LISTDIR=TOTAIL
- LISTPOS=TAIL and LISTDIR=TOHEAD

,LOCBYCURSOR

Use this input parameter with LISTNUM to designate the starting entry to be read. The designated entry is the entry to which the LISTNUM list cursor is pointing.

Be aware that the list cursor could have been reset to zeros by a previous request that, for instance, deleted or moved the entry to which the list cursor points, or updated the list cursor after the last entry on the list had been processed. See *z/OS MVS Programming: Sysplex Services Guide* for more information about using the list cursor.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=COND

Use this input parameter to specify that the lock operation for the lock specified by LOCKINDEX be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

LOCKOPER=NOTHELD

The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released, or your request fails depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

LOCKOPER=HELDBY

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE
Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVR.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,TYPE=ENTDATA
,TYPE=ADJDATA
,TYPE=ECONTROLS
Use this input parameter to specify the type of data to be read from each processed entry.

ENTDATA
Entry data is read.

ADJDATA
Adjunct data is read.

ECONTROLS
List entry control data is read.

You can specify one, two, or all three of these data types, and you can code them in any order on the TYPE parameter. When more than one data type is specified, the format of the parameter would be: TYPE=(ENTDATA,ADJDATA,ECONTROLS). The order in which you code the values does not affect the order in which the system returns the data in the BUFFER area or BUFLIST buffers. See *z/OS MVS Programming: Sysplex Services Guide* for more information about how the data is arranged in the buffer(s).

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of each entry from the starting entry to the head or tail of the list (as specified by LISTDIR). Only those entries with a version that meets the condition specified by the VERSCOMPTYPE parameter are read.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:

- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of successfully processed entries (field LAAREADCNT). • The list entry controls for the first UNPROCESSED entry in the list sequence (field LAALCTL). • The list entry controls for the first PROCESSED entry in the list sequence (field LAARLRMLCTLS). **Action:** Reissue the request beginning with the first unprocessed entry specified as the starting entry. The list entry controls (LAALCTL) will contain the ENTRYID, ENTRYNAME, or ENTRYKEY of the first unprocessed entry. Before you reissue the request, be sure to process all the returned data. For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved and the following fields in the answer area have been filled in: • LAARLRMLCTLS - The first PROCESSED entry in the list sequence. • LAAREADCNT - The number of entries processed successfully • LAALCTL (if the request completed prematurely as well) - The list entry controls for the first UNPROCESSED entry in the list sequence. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro. • If LAALCTL is not zeros, the request completed prematurely. See the action suggested for return code X'4' reason code X'xxxx0409'. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** The request has completed prematurely because the buffer specified by BUFFER or the buffers specified by BUFLIST are full. The following information has been returned in the answer area: • The number of successfully processed entries (field LAAREADCNT). • The list entry controls for the first unprocessed entry in the list sequence (field LAALCTL). • The list entry controls for the first processed entry in the list sequence (field LAARLRMLCTLS). **Action:** Reissue the request beginning with the first unprocessed entry. If the same buffers are used, the data returned from the original request will be overwritten, so be sure to process the returned data first. For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0822	**Equate Symbol:** IXLRSNCODEBADREADTYPE **Meaning:** Program error. You specified that either entry data or adjunct data should be returned, but the list structure does not contain the type of data you requested. No data is returned. **Action:** Ensure that you are requesting the type of data you need from the structure and that the structure supports that type of data. Issue IXLMG to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by ENTRYID or ENTRYNAME exists in the structure, but does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. The answer area (field LAALCTL) contains the list entry controls for the first entry selected for processing. **Action:** If more space is available, specify a larger buffer size, and reissue the request. Consider using the BUFLIST parameter instead of the BUFFER parameter.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 63. Return and Reason Codes for IXLLIST REQUEST=READ_LIST Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=READ_MULT

Description

A READ_MULT request allows you to read multiple entries from the list structure into the storage areas specified by ANSAREA and/or BUFLIST or BUFFER and/or ADJAREA. You can restrict processing to only those entries that satisfy either one or both of the following criteria:
- Those residing on a certain list (which you specify using LISTNUM)
- Those containing data that satisfies a version number comparison (which you specify using VERSCOMP).

For structures allocated in a CFLEVEL=0 coupling facility, if you specify neither LISTNUM nor VERSCOMP, every entry in the structure is read. The order in which the entries are read is unpredictable.

With a coupling facility of CFLEVEL=1 or higher, you can restrict processing to only those entries that satisfy one or more of the following criteria:
- A successful comparison of the list authority value for the target list with a user-specified value. You specify the list authority requirements using AUTHCOMP and AUTHCOMPTYPE.
- A successful comparison of the list entry key value that is being processed with a user-specified value (KEYCOMP).
- A successful comparison of the version number, which is enhanced to include a less-than-or-equal comparison operation.

For each processed entry, any combination of the following types of data can be read depending on what you specify on the TYPE parameter:
- List entry controls
- Entry data
- Adjunct data

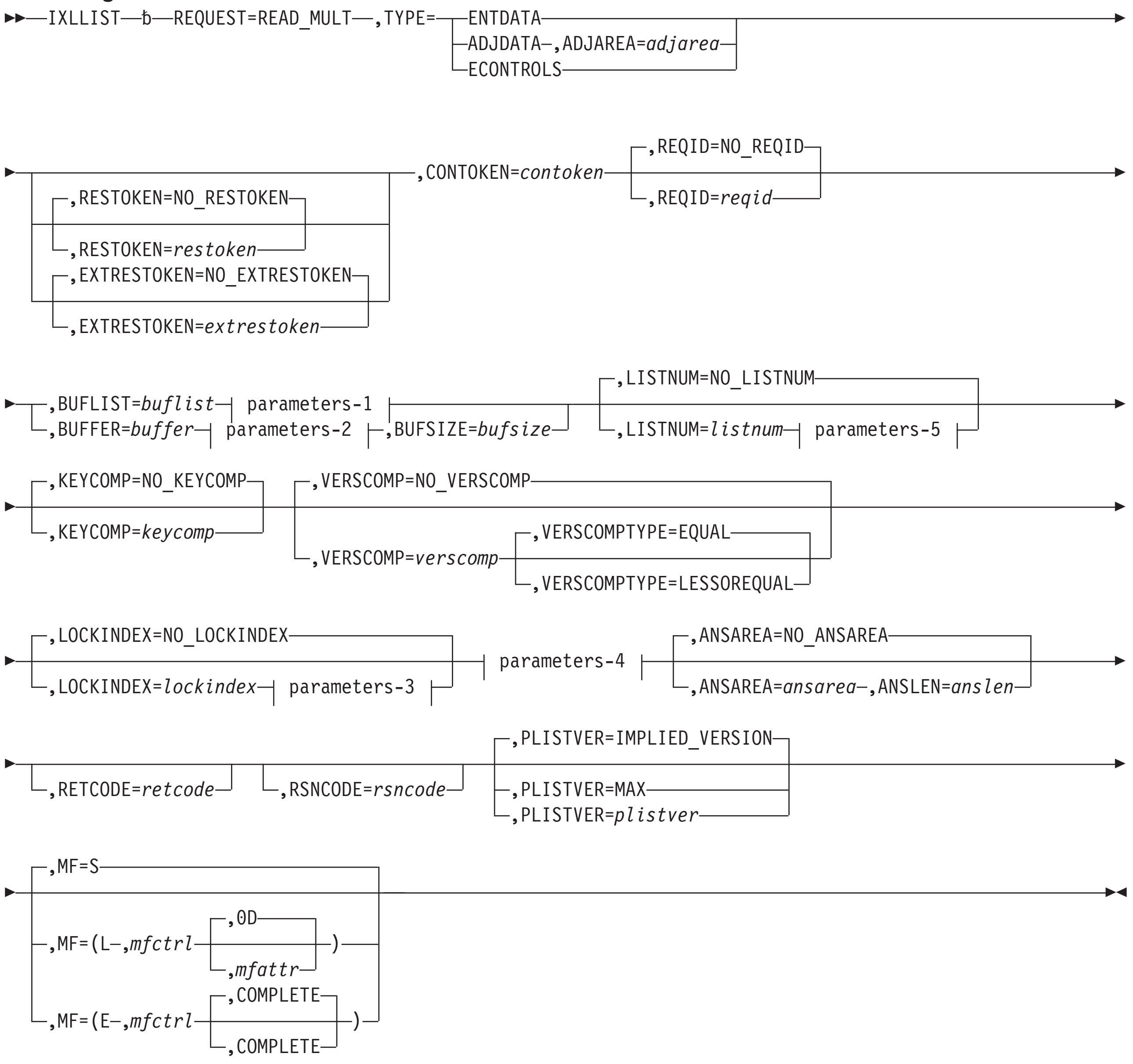
If the request completes prematurely because it exceeds the coupling facility model-dependent time-out criteria or the specified buffer area (BUFFER or BUFLIST) is full, either a restart token (RESTOKEN) or an extended restart token (EXTRESTOKEN) is returned in the answer area (field LAARESTOKEN or LAAEXTRESTOKEN). The token can be specified on the next READ_MULT request to resume processing with the next data item to be processed. Resumed requests are processed identically whether using the RESTOKEN or EXTRESTOKEN to specify the starting location.

Syntax Diagram

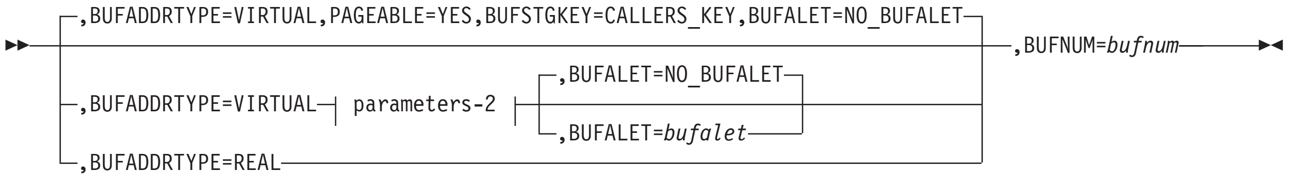
The syntax diagram for IXLLIST REQUEST=READ_MULT is as follows:

IXLLIST Macro

main diagram



parameters-1



parameters-2

```
   ┌─,PAGEABLE=YES,BUFSTGKEY=CALLERS_KEY──────┐
►►─┼──────────────────────────────────────────┼──────────────────────────────────►◄
   │                ┌─,BUFSTGKEY=CALLERS_KEY─┐ │
   ├─,PAGEABLE=YES──┼────────────────────────┼─┤
   │                └─,BUFSTGKEY=bufstgkey───┘ │
   └─,PAGEABLE=NO─────────────────────────────┘
```

parameters-3

```
►►──,LOCKOPER=──┬─NOTHELD─,LOCKMODE=COND──────────────┬──────────────────────────►◄
                │        ┌─,LOCKCOMP=NO_LOCKCOMP─┐    │
                └─HELDBY─┼───────────────────────┼────┘
                         └─,LOCKCOMP=lockcomp────┘
```

parameters-4

```
   ┌─,MODE=SYNCSUSPEND─────────────────────────┐
►►─┼───────────────────────────────────────────┼─────────────────────────────────►◄
   ├─,MODE=SYNCECB─,REQECB=reqecb──────────────┤
   │                ┌─,REQDATA=NO_REQDATA─┐    │
   ├─,MODE=SYNCEXIT─┼─────────────────────┼────┤
   │                └─,REQDATA=reqdata────┘    │
   ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken────────┤
   ├─,MODE=ASYNCECB─,REQECB=reqecb─────────────┤
   │                 ┌─,REQDATA=NO_REQDATA─┐   │
   ├─,MODE=ASYNCEXIT─┼─────────────────────┼───┤
   │                 └─,REQDATA=reqdata────┘   │
   ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken───────┤
   └─,MODE=ASYNCNORESPONSE─────────────────────┘
```

parameters-5

```
   ┌─,AUTHCOMP=NO_AUTHCOMP──────────────────────────────┐
►►─┼────────────────────────────────────────────────────┼────────────────────────►◄
   │                     ┌─,AUTHCOMPTYPE=EQUAL───────┐   │
   └─,AUTHCOMP=authcomp──┼───────────────────────────┼───┘
                         └─,AUTHCOMPTYPE=LESSOREQUAL─┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
2. If MODE=ASYNCNORESPONSE is specified, BUFFER, BUFLIST, and LOCKINDEX may not be specified.

Parameter Descriptions

The parameter descriptions for REQUEST=READ_MULT are listed in alphabetical order. Default values are underlined:

REQUEST=READ_MULT
Use this input parameter to read multiple entries from within the entire structure.

,ADJAREA=*adjarea*
Use this output parameter to specify a storage area to contain the adjunct data that was read from the first processed entry. Adjunct data is returned only if ADJDATA is specified on the TYPE parameter.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area (the adjunct area) to contain the adjunct data.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*

Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion of the request, the answer area contains:

- List entry controls (field LAARLRMLCTLS mapped by IXLYLCTL) from the first processed list entry (if ECONTROLS is specified on the TYPE parameter).
- The number of processed entries (field LAAREADCNT).

If the request completes prematurely, the answer area contains:

- A restart token (LAARESTOKEN) or extended restart token (LAAEXTRESTOKEN) that can be specified on a subsequent READ_MULT request. (See the RESTOKEN and EXTRESTOKEN parameter descriptions.)
- The number of processed entries (LAAREADCNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) where the information from the request will be put.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL

The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL

The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this output parameter to specify a buffer area to hold data that is read from list entries.

You must ensure that the storage area specified by BUFFER:
- Is a multiple of 4096 bytes
- Is less than or equal to 65536 bytes
- Starts on a 4096-byte boundary

  See the BUFSIZE parameter description for specifying the size of the buffer.

The type of data returned to the buffer depends on which types of data you request on the TYPE parameter. See *z/OS MVS Programming: Sysplex Services Guide* for more detailed information about how data is returned from this request.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) where the data read from list entries will be put.

,BUFLIST=*buflist*

Use this output parameter to specify a list of buffers to hold data that is read from list entries. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:
- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  – The left (high-order) four bytes are reserved and
  – The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:
- Reside in the same address space or same data space.
- Be 4096 bytes.
- Start on a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single buffer, even if the buffers are not contiguous.

See the BUFNUM parameter description to specify the number of buffers in the buffer list.

The type of data returned to the buffers depends on which types of data you request on the TYPE parameter. See *z/OS MVS Programming: Sysplex Services Guide* for more detailed information about how data is returned from this request.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*
Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers (0 to 16) in the list (BUFLIST).

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*
Use this input parameter to specify an extended restart token that can be used to resume processing of a READ_MULT request that completed prematurely. The extended restart token is returned in the answer area (field LAAEXTRESTOKEN), and should be specified on the next READ_MULT request to resume processing.

If the request does not complete prematurely, the extended restart token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes list services to treat all of the entries as unprocessed.

2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the extended restart token.

,KEYCOMP=NO_KEYCOMP
,KEYCOMP=*keycomp*

Use this input parameter to specify a value to be compared to the entry key of each list entry in the designated list.

If the list entry that is to be processed does not have an entry key value that is equal to the specified KEYCOMP value, the entry is not processed. Processing continues with the next entry.

Note: The KEYCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the value to be compared to the entry key of the designated list entry.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entries to be read must reside.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=COND
Use this input parameter to specify that the lock operation for the lock specified by LOCKINDEX be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY
Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

LOCKOPER=NOTHELD
The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released, or your request fails depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

LOCKOPER=HELDBY

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro.

For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*
Use this input parameter to specify a restart token that can be used to resume processing of a READ_MULT request that completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The restart token is returned in the answer area (field LAARESTOKEN), and should be specified on the next READ_MULT request to resume processing with the next entry to be read.

If the request does not exceed the model-dependent time-out criteria, the returned token will not be provided.

Notes:

1. Specifying an extended restart token of all zeros causes list services to treat all of the entries as unprocessed.
2. Do not specify an extended restart token other than the one returned in the answer area or one set to all zeros, because results will be unpredictable.

Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the restart token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,TYPE=ENTDATA

,TYPE=ADJDATA
,TYPE=ECONTROLS
Use this input parameter to specify the type of data to be read from each processed entry.

ENTDATA
Entry data is read.

ADJDATA
Adjunct data is read.

ECONTROLS
List entry control data is read.

You can specify one, two, or all three of these data types, and you can code them in any order on the TYPE parameter. When more than one data type is specified, the format of the parameter would be: TYPE=(ENTDATA,ADJDATA,ECONTROLS). The order in which you code the values does not affect the order in which the system returns the data in the BUFFER area or BUFLIST buffers. See *z/OS MVS Programming: Sysplex Services Guide* for more information about how the data is arranged in the buffer(s).

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*
Use this input parameter to specify a version number to be compared to the version number of each entry to be read. Only those entries with a version that meets the condition specified by the VERSCOMPTYPE parameter are read.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING

8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request has completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of successfully processed entries (field LAAREADCNT) • The list entry controls for the first processed entry (field LAARLRMLCTLS). • A token for restarting the request (field LAARESTOKEN or LAAESTRESTOKEN) **Action:** Reissue the request using the restart token (RESTOKEN or EXTRESTOKEN). For more information about premature completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved and the following fields in the answer area have been filled in: • LAARLRMLCTLS - The first PROCESSED entry in the list sequence. • LAAREADCNT - The number of entries processed successfully • LAALCTL (if the request completed prematurely as well) - The list entry controls for the first UNPROCESSED entry in the list sequence. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro. • If LAALCTL is not zeros, the request completed prematurely. See the action suggested for return code X'4' reason code X'xxxx0409'. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** Program error. The request has completed prematurely because the buffer specified by BUFFER or the buffers specified by BUFLIST are full. The following information has been returned in the answer area: • The number of successfully processed entries (field LAAREADCNT). • A restart token (field LAARESTOKEN) or extended restart token (field LAAEXTRESTOKEN). • The list entry controls for the first processed entry in the list sequence (field LAARLRMLCTLS). **Action:** Reissue the request, or increase the size of the buffer(s), and rerun your program. You can reissue the request using the restart token (RESTOKEN) or extended restart token (EXTRESTOKEN). If the same buffers are used, the data returned from the original request will be overwritten so be sure to process the returned data first. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0822	**Equate Symbol:** IXLRSNCODEBADREADTYPE **Meaning:** Program error. You specified that either entry data or adjunct data should be returned, but the list structure does not contain the type of data you requested. No data is returned. **Action:** Ensure that you are requesting the type of data you need from the structure and that the structure supports that type of data. Issue IXLMG to get more information about the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to restart the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request (field LAARESTOKEN). • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. The answer area (field LAALCTL) contains the list entry controls for the first entry selected for processing. **Action:** If more space is available, specify a larger buffer size, and reissue the request. Consider using the BUFLIST parameter instead of the BUFFER parameter.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST and BUFNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST. • Buffer boundaries.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be stored in the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. **Action:** Reinitiate the request with an EXTRESTOKEN value of zero. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. Discard the results of the initial request and reissue the request. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 64. Return and Reason Codes for IXLLIST REQUEST=READ_MULT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=WRITE

Description

A WRITE request writes entry data and/or adjunct data to an existing or new list entry depending on the value specified for ENTRYTYPE. Entry data is written to the **data entry** segment of a list entry, while the adjunct data is written to the **adjunct data area** segment. Before issuing the request, the data to be written to the entry should be stored as follows:

- Use the BUFFER or BUFLIST parameter to identify the storage containing the data to be placed in the data entry.
- Use the ADJAREA parameter to identify the storage containing the data to be placed in the adjunct area.

Note: If the structure supports entry data and BUFFER and BUFLIST are not specified, then the created list entry does not contain any entry data. If the structure supports adjunct data and ADJAREA is not specified, then the created list entry contains binary zeros in the adjunct area.

If you are writing entry data to the structure, you can specify the number of data elements comprising the data entry segment of an existing or new entry by using ELEMNUM.

Additionally, you can perform locking functions with a WRITE request by specifying LOCKOPER. The lock designated by LOCKINDEX will be operated on as specified by LOCKOPER.

With a coupling facility of CFLEVEL=1 or higher, the following additional functions are supported for a WRITE request:

- You can specify that a list entry is to be written only after a successful comparison of the list authority value for the target list with a user-specified value. You specify the list authority comparison requirements using AUTHCOMP and AUTHCOMPTYPE. If the request is successful, you also can update the list authority value to a new value that you specify with NEWAUTH.
- Several options are available for updating the list cursor, based on request completion.
- Version number comparison is enhanced to allow for an additional equal-or-less-than-equal comparison operation.
- The system can assign an entry key value automatically from a list control list key value.

When the request successfully completes, list entry controls, the number of elements or entries residing on the list, and the total number of allocated entries in the structure are returned in the answer area (ANSAREA) if specified (mapped by IXLYLAA).

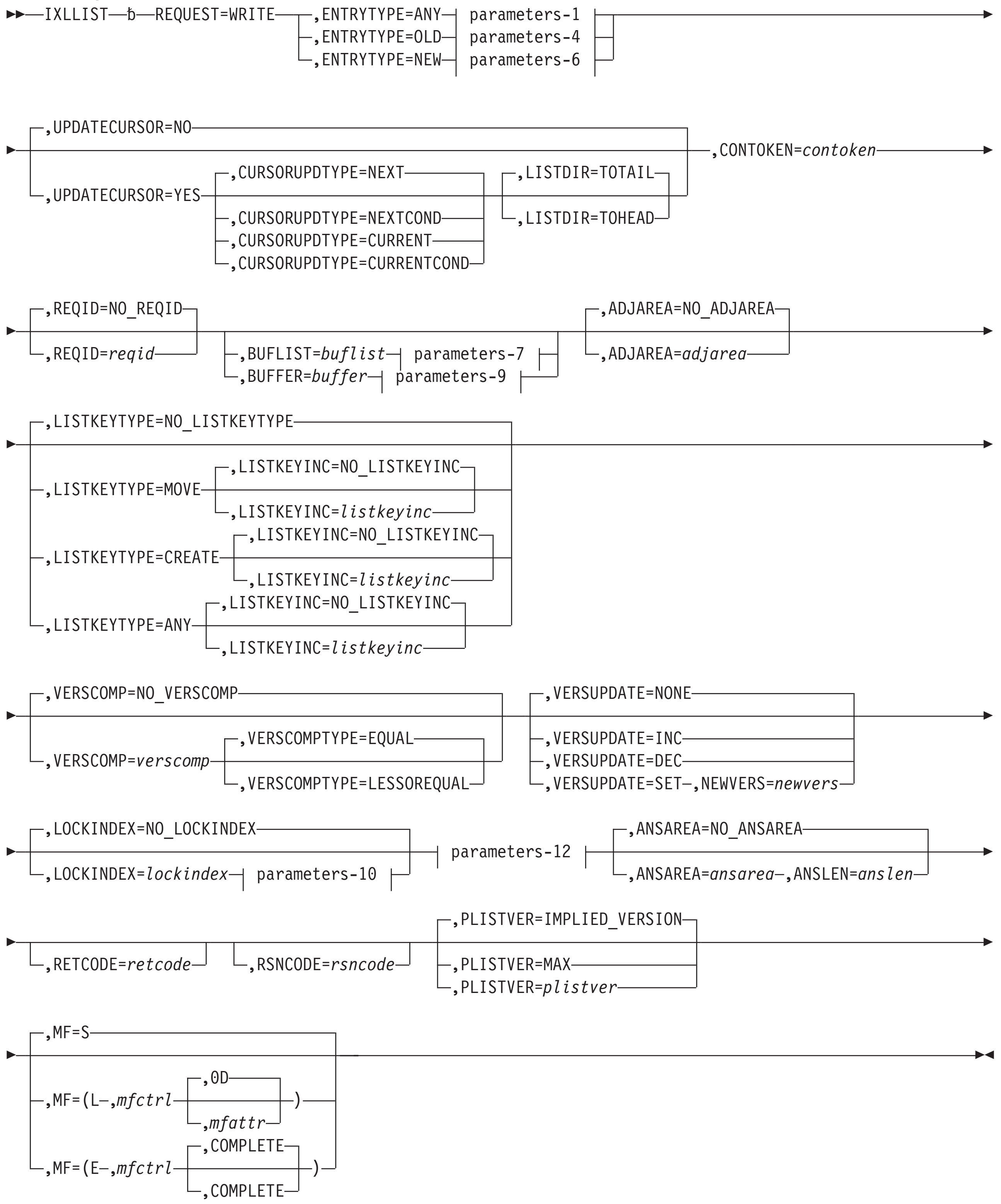
See *z/OS MVS Programming: Sysplex Services Guide* for more information on how to use the IXLLIST REQUEST=WRITE request.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=WRITE is as follows:

IXLLIST Macro

main diagram



parameters-1

```
                        ┌─,ENTRYKEY=NO_ENTRYKEY──────────────────────────────┐
►►──┬─,LISTNUM=listnum──┼────────────────────────────────────────────────────┼──┬──────────►
    │                   │                       ┌─,KEYREQTYPE=EQUAL──────────┐ │  │
    │                   └─,ENTRYKEY=entrykey────┼────────────────────────────┼─┘  │
    │                                           ├─,KEYREQTYPE=LESSOREQUAL────┤    │
    │                                           └─,KEYREQTYPE=GREATEROREQUAL─┘    │
    ├─,ENTRYID=entryid─┤ parameters-2 ├───────────────────────────────────────────┤
    ├─,ENTRYNAME=entryname─,LISTNUM=listnum───────────────────────────────────────┤
    └─,LOCBYCURSOR─┤ parameters-3 ├───────────────────────────────────────────────┘

  ┌─,LISTPOS=HEAD──────┐
►─┼────────────────────┼──────────────────────────────────────────────────────────►◄
  ├─,LISTPOS=TAIL──────┤
  └─┤ parameters-13 ├──┘
```

parameters-2

```
►►──,LISTNUM=listnum──┬──────────────────────────┬───────────────────────────────►◄
                      ├─,ENTRYKEY=entrykey───────┤
                      └─,ENTRYNAME=entryname─────┘
```

parameters-3

```
►►──,LISTNUM=listnum──┬──────────────────────────┬───────────────────────────────►◄
                      ├─,ENTRYKEY=entrykey───────┤
                      └─,ENTRYNAME=entryname─────┘
```

parameters-4

```
►►──┬─,LISTNUM=listnum─┤ parameters-5 ├─┤ parameters-13 ├─┬──────────────────────►◄
    │                   ┌─,LISTNUM=NO_LISTNUM─┐           │
    ├─,ENTRYID=entryid──┼─────────────────────┼───────────┤
    │                   └─,LISTNUM=listnum────┘           │
    │                       ┌─,LISTNUM=NO_LISTNUM─┐       │
    ├─,ENTRYNAME=entryname──┼─────────────────────┼───────┤
    │                       └─,LISTNUM=listnum────┘       │
    └─,LOCBYCURSOR─,LISTNUM=listnum───────────────────────┘
```

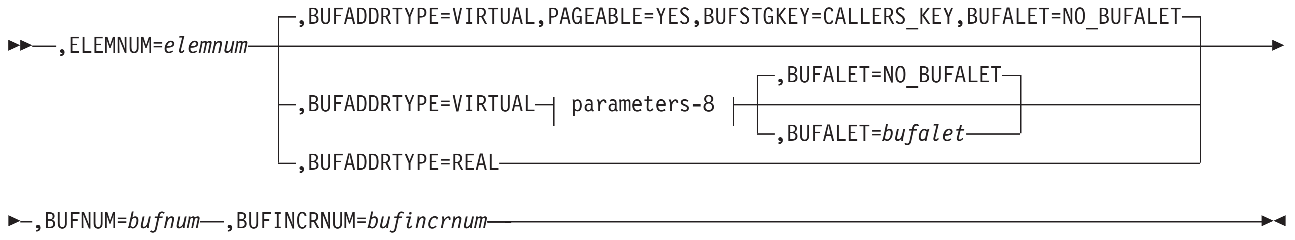
parameters-5

```
    ┌─,LISTPOS=HEAD─┐   ┌─,ENTRYKEY=NO_ENTRYKEY──────────────────────────────┐
►►──┼───────────────┼───┼────────────────────────────────────────────────────┼───►◄
    └─,LISTPOS=TAIL─┘   │                       ┌─,KEYREQTYPE=EQUAL──────────┐ │
                        └─,ENTRYKEY=entrykey────┼────────────────────────────┼─┘
                                                ├─,KEYREQTYPE=LESSOREQUAL────┤
                                                └─,KEYREQTYPE=GREATEROREQUAL─┘
```

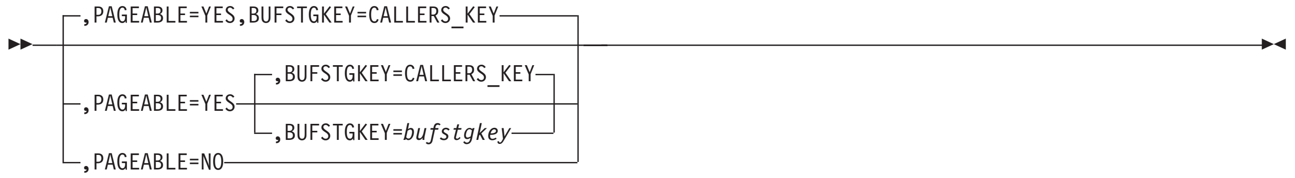
parameters-6

```
                                                  ┌─,LISTPOS=HEAD─┐
►►──,LISTNUM=listnum──┬──────────────────────────┬─┼───────────────┼──────────────►◄
                      ├─,ENTRYNAME=entryname─────┤ └─,LISTPOS=TAIL─┘
                      └─,ENTRYKEY=entrykey───────┘
```

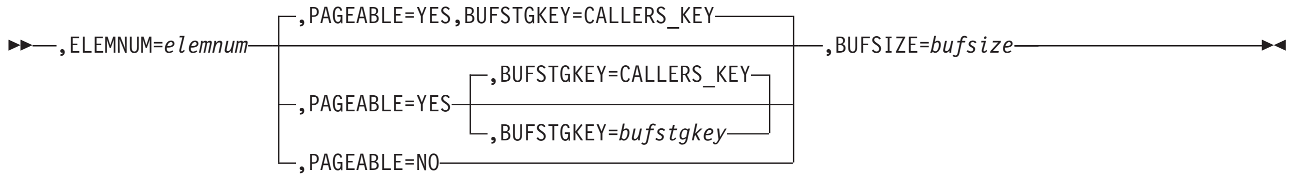
parameters-7



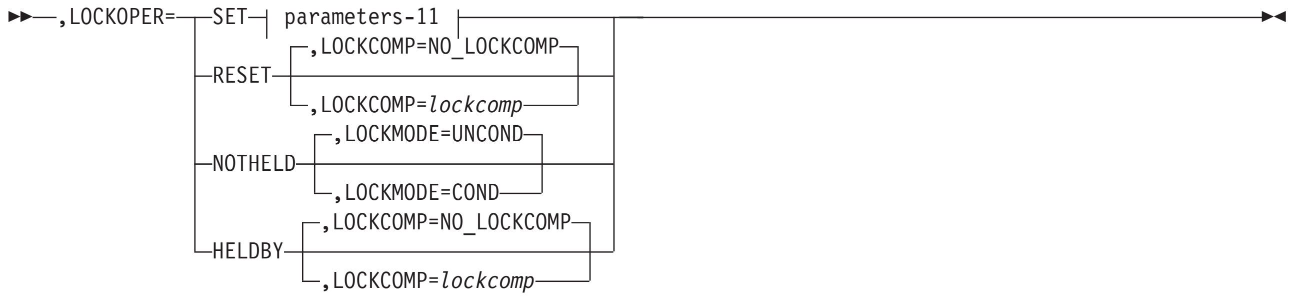
parameters-8



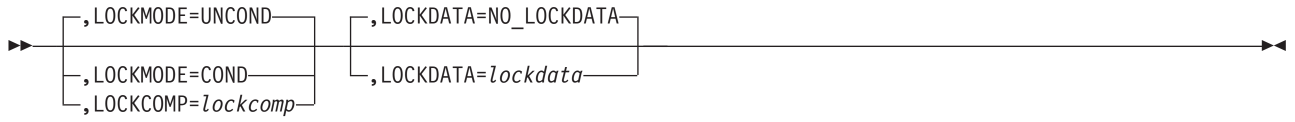
parameters-9



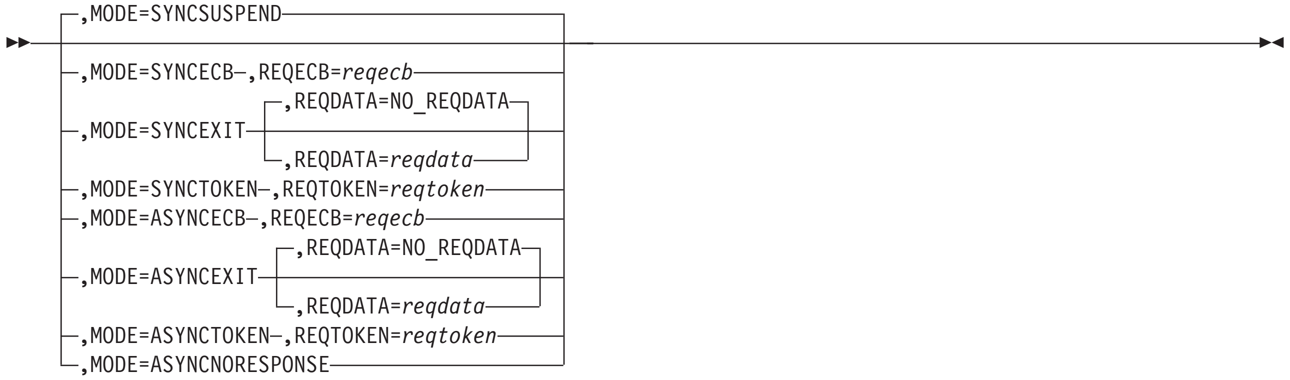
parameters-10



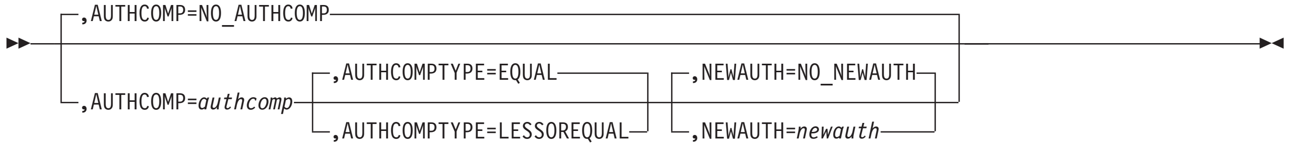
parameters-11



parameters-12


,MODE=SYNCSUSPEND
,MODE=SYNCECB ,REQECB=*reqecb*
,MODE=SYNCEXIT ,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
,MODE=SYNCTOKEN ,REQTOKEN=*reqtoken*
,MODE=ASYNCECB ,REQECB=*reqecb*
,MODE=ASYNCEXIT ,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
,MODE=ASYNCTOKEN ,REQTOKEN=*reqtoken*
,MODE=ASYNCNORESPONSE

parameters-13



Notes:

1. If ENTRYTYPE is not specified, ENTRYTYPE=ANY is the default and you must code the required parameters shown in the | parameters-1 | fragment.
2. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
3. If MODE=ASYNCNORESPONSE is specified, you may not code BUFFER, BUFLIST, or LOCKINDEX.

Parameter Descriptions

The parameter descriptions for REQUEST=WRITE are listed in alphabetical order. Default values are underlined:

REQUEST=WRITE
Use this input parameter to write data to a list entry.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*
Use this input parameter to specify a storage area to contain the adjunct data to be written to the existing or new entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area that contains the adjunct data.

,ANSAREA=NO_ANSAREA
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

On a successful completion (for a synchronous request: RC=0, for an asynchronous request: ECB is posted, IXLFCOMP indicates completion, or your complete exit ran) of the request, the following information is returned to the answer area:

- List entry controls of the existing or new entry (field LAALCTL).

- The number of list entries or elements residing on the list (field LAALISTCNT).
- The total number of allocated entries in the structure (field LAATOTALCNT).
- The total number of allocated elements in the structure (field LAATOTALELECNT).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned by the request.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input parameter to specify a buffer area to hold the entry data to be written to the existing or new entry.

You can define the buffer size to be a total of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

See the BUFSIZE parameter description for defining the size of the buffer.

Note: You cannot code BUFFER with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry data.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this input parameter to specify a list of buffers to hold the entry data to be written to the existing or new entry. BUFLIST specifies a 128-byte storage area that consists of a list of 0 to 16 buffer addresses.

The 128-byte storage area must:

- Consist of 0 to 16 elements.
- Each element consists of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:

- Reside in the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. List services treats BUFLIST buffers as a single, contiguous buffer, even if the buffers are not contiguous.

See BUFNUM and BUFINCRNUM parameter descriptions for defining the number and size of buffers.

Note: You cannot code BUFLIST with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains a list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be written to the list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CURSORUPDTYPE=NEXT
,CURSORUPDTYPE=NEXTCOND
,CURSORUPDTYPE=CURRENT
,CURSORUPDTYPE=CURRENTCOND

Use this input parameter to specify how the list cursor is to be updated when UPDATECURSOR=YES is specified.

Note: The NEXTCOND, CURRENT, and CURRENTCOND values are valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

CURSORUPDTYPE=NEXT

Update the list cursor to point to the list entry before or after the target entry. The direction of the cursor update depends on the value specified for LISTPOS or LISTDIR.

CURSORUPDTYPE=NEXTCOND

Update the list cursor to point to the list entry before or after the target entry only if the target list entry is moved or deleted.

You set the list cursor direction (so that it will point to either before or after the target entry) with SETCURSOR on a WRITE_LCONTROLS request.

The list cursor will be reset to binary zeros if either:

- The entry is the last entry on the list and the list cursor direction is set to a head-to-tail direction.
- The entry is the first entry on the list and the list cursor is set to a tail-to-head direction.

CURSORUPDTYPE=CURRENT
Set the list cursor to point to the list entry processed for the request.

If the list entry is deleted or moved to another list, then the list cursor will be reset to binary zeros.

CURSORUPDTYPE=CURRENTCOND
Set the list cursor to point to the list entry processed only if the list cursor value currently is zero and the target list entry is not deleted or moved to another list.

If the list entry is deleted or moved to another list, then the list cursor will remain binary zeros.

,ELEMNUM=*elemnum*
Use this input parameter to specify the number of elements to be allocated to the existing or new data entry. Valid ELEMNUM values are from zero to the MAXELEMNUM value that was specified on the IXLCONN macro.

Considerations for specifying ELEMNUM are:

- If this request creates a new entry, the number of elements is set to the ELEMNUM value.
- If the entry already exists, the number of elements is updated to the ELEMNUM value specified.

Note: If the data from the buffers exceeds the amount of space in the elements, the data will be truncated. If the data from the buffers is less than the amount of space in the elements, the remaining space will be padded with binary zeros.

If you specify an ELEMNUM value of zero:

- No entry data will be written to the new entry.
- Any entry data in the existing entry will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of elements.

,ENTRYID=*entryid*
Use this input parameter to designate the entry identifier of an existing entry to be updated. ENTRYID applies only to locating an existing entry; it does not determine the identifier of a new entry should one be created.

If an entry with this identifier does not exist (and ENTRYTYPE=ANY or ENTRYTYPE=NEW is specified) a new entry is created on the LISTNUM list with a unique entry identifier assigned by list services. The new entry identifier is returned in the answer area in the field LCTLENTRYID in LAALCTL. (The entry identifier you specify will be ignored.)

When ENTRYID is specified with ENTRYNAME, ENTRYKEY, or LISTPOS, list services uses only ENTRYID to determine if the entry already exists.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=NO_ENTRYKEY
,ENTRYKEY=*entrykey*
Use this input parameter to either:

- Designate the entry key of an existing entry to be updated.
- Assign an entry key of your choice to a new entry.

The existing entry must reside on the LISTNUM list; the new entry will be created on the LISTNUM list.

Whether the specified entry key designates an existing entry or is assigned to a new entry depends on what ENTRYTYPE value you specify and whether the entry already exists.

If there is a sublist of entries on the list all with the same key as the ENTRYKEY key (or that satisfies the KEYREQTYPE criteria, if specified), the LISTPOS parameter determines which entry in the sublist (the HEAD or TAIL entry) is updated, or, if an entry is created, at which end of the sublist (the HEAD or TAIL) the new entry is positioned.

If this request creates a new entry, but ENTRYKEY is not specified (and the structure supports keyed entries), list services assigns a default key value to the new entry as follows:
- If LISTPOS=HEAD, the new list entry key is set to all binary zeros.
- If LISTPOS=TAIL, the new list entry key is set to all binary ones.

Specify ENTRYKEY only for structures that support keyed entries.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*

Use this input parameter to either:
- Designate the entry name of an existing entry to be updated.
- Assign an entry name to a new entry to be created.

The new entry will be created on the LISTNUM list. The existing entry can reside on any list.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,ENTRYTYPE=ANY
,ENTRYTYPE=OLD
,ENTRYTYPE=NEW

Use this input parameter to specify whether you want to create a new entry, update an existing entry, or either.

ANY

Either update an existing entry or create a new entry. If the designated entry exists, its data is replaced with the new data. If the designated entry does not exist, one will be created with the new data.

OLD

Update an existing entry. The designated entry must already exist, or the request fails.

NEW

Create a new entry.
- If you specify ENTRYNAME, a new entry is created only if the entry does not already exist. If the entry exists, the request fails.
- If you specify ENTRYKEY, a new entry will be created regardless of whether another entry with the same key already exists on the designated list.
- If you specify LISTPOS without ENTRYNAME or ENTRYKEY, a new entry is created at the specified HEAD or TAIL of the list.

Note: When a new entry is created, the ENTRYID is returned in the answer area in the LCTLENTRYID field of LAALCTL.

,KEYREQTYPE=EQUAL

,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL

Use this input parameter to specify how, if at all, the entry key of the existing entry to be updated can differ from the entry key specified by ENTRYKEY. KEYREQTYPE applies only to locating an existing entry; it does not determine the key of a new entry should one be created.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

Notes:

1. When no entries on the list meet the requirements of the KEYREQTYPE, and a new entry cannot be created (ENTRYTYPE=OLD was specified), the request will be failed with a return code X'8' and reason code IXLRSNCODENOENTRY, or a new list entry will be allocated, depending on the other options specified (ENTRYTYPE=NEW or ENTRYTYPE=ANY).
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, the entry with an entry key value closest to the value specified will be selected. When multiple entries have the same entry key value, LISTPOS is used to resolve whether the first or last entry with the entry key value is selected.

,LISTDIR=TOTAIL
,LISTDIR=TOHEAD

Use this input parameter with UPDATECURSOR to specify the direction in which the list cursor is moved as a result of this request. See the UPDATECURSOR parameter for a full description of LISTDIR.

,LISTKEYINC=NO_LISTKEYINC
,LISTKEYINC=*listkeyinc*

Use this input parameter to specify a value to be added to the list key after the entry key is set to the list key value. If the entry key is not set to the list key value, the list key value will not be changed. If the result of adding the value specified by LISTKEYINC to the list key value is greater than the maximum list key value, the system fails the request.

Note: The LISTKEYINC parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value to be added to the list key after the entry key is set to the list key value.

LISTKEYTYPE=NOLISTKEY
LISTKEYTYPE=CREATE
LISTKEYTYPE=ANY

Use this input parameter to specify when the designated entry key is to be set to the current list key value. You can set the entry key to the current list key value when you create the entry.

(Set the list key and maximum list key values with a WRITE_LCONTROLS request.)

Note: The LISTKEYTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

LISTKEYTYPE=NOLISTKEY
Do not set the entry key for the target list entry to the current list key value. If the list entry is

created and the list structure supports entries with keys, then the other parameters specified (such as ENTRYKEY) will determine the resultant entry key value.

LISTKEYTYPE=CREATE
Set the entry key for the designated list entry to the current list key value if the entry is created by this request.

LISTKEYTYPE=ANY
Set the entry key for the designated list entry to the current list key value if the entry is created by this request. (This parameter also applies when you MOVE an entry.)

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*
Use this input parameter to specify the number of the list to which the data will be written. Use LISTNUM in the following ways:

- With LISTPOS and/or ENTRYKEY to designate an existing entry to be updated, or a position on the list for a new entry.
- With ENTRYID or ENTRYNAME to:
  - Check if the entry exists on the list (when ENTRYTYPE=ANY) before proceeding with the request.
  - Confirm that the entry exists on the list (when ENTRYTYPE=OLD) before proceeding with the request.
  - Confirm that the entry does not exist on the list (when ENTRYTYPE=NEW) before proceeding with the request.
- With LOCBYCURSOR to designate an existing entry to be updated.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTPOS=HEAD
,LISTPOS=TAIL
Use this input parameter to designate either:

- The position on the LISTNUM list where the new entry is placed (if this request creates a new entry).
- The existing entry that is to be updated (if this request updates an existing entry).

LISTPOS designates the position or entry at the HEAD or the TAIL of a list or sublist:

- If you specify ENTRYKEY to designate the position or entry and the list contains a sublist of one or more entries with the same key (or that satisfies the KEYREQTYPE criteria, if specified), then:
  - LISTPOS=HEAD designates the position or entry at the head of the sublist.
  - LISTPOS=TAIL designates the position or entry at the tail of the sublist.
- If you do not specify ENTRYKEY to designate the position or entry, then:
  - LISTPOS=HEAD designates the position or entry at the head of the list.
  - LISTPOS=TAIL designates the position or entry at the tail of the list.

,LOCBYCURSOR
Use this input parameter to designate an existing entry to be updated. The designated entry is the entry which is identified by the list cursor for a particular list (LISTNUM).

Be aware that the list cursor could have been reset to zeros by a previous request that, for instance, deleted or moved the entry to which the list cursor points, or updated the list cursor after the last entry on the list had been processed. See *z/OS MVS Programming: Sysplex Services Guide* for more information about using the list cursor.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*
This parameter has slightly different meanings based on the value specified for the LOCKOPER

parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA
,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:
- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKINDEX=NO_LOCKINDEX
,LOCKINDEX=*lockindex*

Use this input parameter to specify the index of the lock to be operated on as specified by LOCKOPER. The lock indexes begin with zero.

Note: You cannot code LOCKINDEX with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND

Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND

If the specified lock is held by another user, your request is either:
- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND

The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET
Set the lock.

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released or fails, depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

RESET
Reset the lock.

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

NOTHELD
The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released or your request fails, depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

HELDBY
Processing is determined by which connector is holding the lock.

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.
- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:
- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND

The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB

The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT

The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN

The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB

The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT

The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN

The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE

The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*

Use this input parameter to specify a list authority value for the list specified by LISTNUM. If NEWAUTH is omitted, the list authority for the designated list is unchanged.

When the structure is allocated, the authority value for each list is initialized to binary zeros.

Note: The NEWAUTH parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher, except on WRITE_LCONTROLS requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,NEWVERS=*newvers*

Use this input parameter with VERSUPDATE=SET to specify a version number to either replace the version number of the existing entry, or initialize the version number of a new entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the entry version number.

,PAGEABLE=YES
,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST are in pageable or potentially pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage. XES performs the required page fixing to fix the buffers in real storage while the cache or list request transfers data to or from the coupling facility.

This includes storage obtained from pageable subpools, disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) This does not include implicitly non-pageable storage (for example, storage obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage. XES does not page fix the buffers in real storage.

This includes implicitly non-pageable storage areas (for example, storage obtained from non-pageable subpools), and may include storage that has the potential to become pageable during the processing of a request (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a cache or list request.)

The system takes responsibility for managing binds to central storage for the duration of the cache or list request, if and only if the non-pageable storage is owned by either the requestor’s address space or the connector’s address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of an IXLCACHE or IXLLIST request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,UPDATECURSOR=NO
,UPDATECURSOR=YES
Use this input parameter to specify whether the list cursor, for the list specified by LISTNUM, is to be updated to point to another entry on the list.

NO
The list cursor is not updated.

YES
The list cursor is updated according to the specification of the CURSORUPDATE keyword.

The list cursor is set to binary zeros if its new position would be before the first entry or after the last entry in the list.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*
Use this input parameter to specify a version number to be compared to the version number of the existing entry to be updated. If this request creates a new entry, the VERSCOMP specification is ignored.

The entry is updated only if its version number meets the condition specified by the VERSCOMPTYPE parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

,VERSUPDATE=NONE
,VERSUPDATE=INC
,VERSUPDATE=DEC
,VERSUPDATE=SET
Use this input parameter to specify how the version number of the existing entry will be updated, or, for those cases where a new entry is created, initialized.

VERSUPDATE=NONE
The existing entry’s version number is not updated. The new entry’s version number is initialized to binary zeros.

VERSUPDATE=INC
The existing entry’s version number is increased by one. The new entry’s version number is initialized to binary zeros, except for the low-order bit, which is set to one.

VERSUPDATE=DEC
The existing entry’s version number is decreased by one. The new entry’s version number is initialized to all binary ones.

VERSUPDATE=SET
Both the existing and the new entry’s version number is set to the value specified by NEWVERS.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA) if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified any of the following, the request could not be completed successfully because the specified lock is not currently held as required: • LOCKOPER=SET or LOCKOPER=NOTHELD with either of: – LOCKMODE=COND – LOCKCOMP • LOCKOPER=HELDBY **Action:** Retry the request, or obtain the lock as required, and retry the request.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.
8	xxxx0837	**Equate Symbol:** IXLRSNCODEBADWRITEADJDATA **Meaning:** Program error. The request specified that adjunct data was to be written, but the area specified by ADJAREA is not addressable. There was no change to the structure. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx083E	**Equate Symbol:** IXLRSNCODEMAXLISTKEY **Meaning:** Program error. The list key to be assigned to an entry that was being created or moved was not valid. Either the list key or the list key plus the list key increment value was greater than the maximum list key. **Action:** Ensure that you specified a correct list key increment. Depending on the protocol you are using for assigning list key values, you might want to issue a WRITE_LCONTROLS request to update either the list key value to a lower value or the maximum list key value to a higher value.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The designated entry has a version number that failed the comparison specified by VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the value specified in VERSCOMP and look at the version returned in the answer area.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by ENTRYID or ENTRYNAME exists in the structure, but does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0841	**Equate Symbol:** IXLRSNCODEBADENTRYNAME **Meaning:** Program error. Entry creation was attempted, but no entry was created for one of the following reasons: • No entry name was specified (and the structure supports names). • The specified entry name is not unique within the structure. The list entry controls for the first entry on the list are returned in the answer area (field LAALCTL). **Action:** Be sure to provide a unique entry name when creating entries to be written to structures that support entry names.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND, or LOCKOPER=NOTHELD with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLIST request type either required the structure to support entry keys or designated a list entry or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect or BUFSTGKEY was not specified and the PSW key does not match the key of the buffers. The data cannot be fetched from the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The value specified for ELEMNUM is not valid. **Action:** Correct the ELEMNUM specification to be within the allowable range: from zero to whatever MAXELEMNUM value was returned to the connector in the connect answer area.

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The request attempted to create a new entry or overwrite an existing entry, but the structure is full and cannot accommodate any more entries. **Action:** Determine why the structure is full. You should be monitoring the usage of the structure every time a read or write is done (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure). This data should be evaluated periodically to ensure structure resources are being used efficiently. You might be able to delete existing entries to free up space, rebuild the structure to make it larger if rebuild is allowed (IXLCONN macro, ALLOWREBLD parameter), or alter the size of the structure (IXLALTER macro).

Table 65. Return and Reason Codes for IXLLIST REQUEST=WRITE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C18	**Equate Symbol:** IXLRSNCODELISTFULL **Meaning:** Environmental error. The entry could not be placed on the designated target list because the list is full. **Action:** Either change the maximum number of entries allowed on the list by specifying a new LISTLIMIT on a WRITE_LCONTROLS request. You should be monitoring the usage of the list structure every time a read or write is done. (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure.) This data should be evaluated periodically to ensure structure resources are being used efficiently.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLIST REQUEST=WRITE_LCONTROLS

Description

A WRITE_LCONTROLS request allows you to modify the list controls for the list specified by LISTNUM. Any one or more of the following list attributes can be changed:

- The list authority (NEWAUTH).
- The list limit (LISTLIMIT) that defines the number of elements or entries allowed on the list.
- The list description (LISTDESC) that you define for the list.

With a coupling facility of CFLEVEL=1 or higher, you can also modify the following additional list controls for the list specified by LISTNUM.

- The list key (LISTKEY) and the maximum value for the list key associated with the list (MAXLISTKEY).
- The list cursor and the direction in which processing with the cursor is to occur (SETCURSOR). The direction you specify applies only when a request to update the list cursor specifies CURSORUPDTYPE=NEXTCOND.

An authority comparison is required for all WRITE_LCONTROLS requests. If you do not specify AUTHCOMP, the system uses a default value of binary zeros for the authority comparison.

Syntax Diagram

The syntax diagram for IXLLIST REQUEST=WRITE_LCONTROLS is as follows:

main diagram

```
                                  ┌─,AUTHCOMP=0────────────┐     ┌─,NEWAUTH=NO_NEWAUTH─┐
►►──IXLLIST──b──REQUEST=WRITE_LCONTROLS─┴─,AUTHCOMP=authcomp─┴──┬──┴─,NEWAUTH=newauth───┴──┬──────►
                                                                  │ ┌─,LISTLIMIT=NO_LISTLIMIT─┐ │
                                                                  ├─┴─,LISTLIMIT=listlimit───┴─┤
                                                                  │ ┌─,LISTLIMIT=NO_LISTLIMIT─┐ │
                                                                  ├─┴─,LISTDESC=listdesc─────┴─┤
                                                                  │ ┌─,LISTKEY=NO_LISTKEY─┐     │
                                                                  ├─┴─,LISTKEY=listkey────┴─────┤
                                                                  │ ┌─,MAXLISTKEY=NO_MAXLISTKEY─┐│
                                                                  ├─┴─,MAXLISTKEY=maxlistkey───┴┤
                                                                  │ ┌─,SETCURSOR=HEAD─┐          │
                                                                  └─┴─,SETCURSOR=TAIL─┴──────────┘

                        ┌─,REQID=NO_REQID─┐
►─,CONTOKEN=contoken────┴─,REQID=reqid────┴─,LISTNUM=listnum──────────────────────────────────────►

                  ┌─,ANSAREA=NO_ANSAREA──────────────┐
►─┤ parameters-1 ├─┴─,ANSAREA=ansarea─,ANSLEN=anslen─┴──┬──────────────────┬──┬──────────────────┬──►
                                                          └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
```

```
   ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S──────────────────────────────┐
►──┼───────────────────────────┼───┼────────────────────────────────────┼───────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤   │                   ┌─,0D──────┐     │
   └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl────┼──────────┼─)───┤
                                   │                   └─,mfattr──┘     │
                                   │                   ┌─,COMPLETE─┐    │
                                   └─,MF=(E─,mfctrl────┼───────────┼─)──┘
                                                       └─,COMPLETE─┘
```

parameters-1

```
     ┌─,MODE=SYNCSUSPEND─────────────────────────┐
►►───┼───────────────────────────────────────────┼──────────────────────────────────────────►◄
     ├─,MODE=SYNCECB─,REQECB=reqecb──────────────┤
     │                ┌─,REQDATA=NO_REQDATA─┐    │
     ├─,MODE=SYNCEXIT─┼─────────────────────┼────┤
     │                └─,REQDATA=reqdata────┘    │
     ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken────────┤
     ├─,MODE=ASYNCECB─,REQECB=reqecb─────────────┤
     │                 ┌─,REQDATA=NO_REQDATA─┐   │
     ├─,MODE=ASYNCEXIT─┼─────────────────────┼───┤
     │                 └─,REQDATA=reqdata────┘   │
     ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken───────┤
     └─,MODE=ASYNCNORESPONSE─────────────────────┘
```

Notes:

1. If MODE=SYNCTOKEN or MODE=ASYNCTOKEN is specified, ANSAREA=*ansarea* and ANSLEN=*anslen* are required.
2. At least one of NEWAUTH, LISTLIMIT, and LISTDESC must be specified. More than one of these parameters may be specified as well.

Parameter Descriptions

The parameter descriptions for REQUEST=WRITE_LCONTROLS are listed in alphabetical order. Default values are underlined:

REQUEST=WRITE_LCONTROLS
Use this input parameter to change the list controls for a specified list.

,ANSAREA=<u>NO_ANSAREA</u>
,ANSAREA=*ansarea*
Use this output parameter to specify an answer area to contain information returned from a failed request. The format of the answer area is described by mapping macro IXLYLAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a size of ANSLEN) where the information about the request will be returned.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=<u>0</u>
,AUTHCOMP=*authcomp*
Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM.

The AUTHCOMP value must equal the list authority of the LISTNUM list, or the request terminates with no change to the structure.

Note: At the time the structure was allocated, the list authority value for each list in the structure was initialized to binary zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the list comparison authority value, or take the default, AUTHCOMP=0.

,CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLIST invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,LISTDESC=*listdesc*

Use this input parameter to specify a description to be associated with the list specified by LISTNUM. The description can be anything you want.

If LISTDESC is not specified, the list description of the LISTNUM list remains unchanged.

Note: At the time the structure was allocated, the list description for each list in the structure was initialized to binary zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte field that contains the list description.

,LISTKEY=NO_LISTKEY
,LISTKEY=*listket*

Use this input parameter to specify the list key to be associated with the list. You may assign a list key value to list entries when they are created or moved.

If LISTKEY is not specified, the list key for the designated list remains unchanged.

Note: At the time the structure was allocated, the list key for each list in the structure was initialized to binary zeros. Use this parameter only with a coupling facility of CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the list key to be associated with the list.

,LISTLIMIT=*listlimit*

Use this input parameter to specify the list limit—the maximum allowable number of entries or elements that can reside on the list specified by LISTNUM. Whether the limit is counted in terms of entries or elements depends on what you specified for the LISTCNTLTYPE parameter on the IXLCONN macro.

If LISTLIMIT is not specified, the list limit of the LISTNUM list remains unchanged.

Note: At the time the structure was allocated, the list limit for each list in the structure was initialized to the total number of entries or elements in the structure, depending on the value specified for LISTCNTLTYPE on the IXLCONN macro.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the list limit.

,LISTNUM=*listnum*
Use this input parameter to specify the number of the list to which the list controls specified by this request will apply.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,MAXLISTKEY=NO_MAXLISTKEY
,MAXLISTKEY=*maxlistkey*
Use this input parameter to specify the maximum list key value to be associated with the list.

Note: At the time the structure was allocated, the maximum list key value for each list in the structure was initialized to binary zeros. Use this parameter only with a coupling facility of CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the maximum list key value.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB

,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify:

- Whether the request is to be performed synchronously or asynchronously
- How you wish to be notified of request completion

MODE=SYNCSUSPEND
The request is suspended until it can complete synchronously. To use this option, your program must be enabled for I/O and external interrupts.

MODE=SYNCECB
The request processes synchronously if possible. If the request processes asynchronously, the ECB specified by REQECB is posted when the request completes.

MODE=SYNCEXIT
The request processes synchronously if possible. If the request processes asynchronously, your complete exit is given control when the request completes.

MODE=SYNCTOKEN
The request processes synchronously if possible. If the request processes asynchronously, an asynchronous request token is returned in the area specified by REQTOKEN on invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=SYNCTOKEN is specified.

MODE=ASYNCECB
The request processes asynchronously. The ECB specified by REQECB is posted when the request completes.

MODE=ASYNCEXIT
The request processes asynchronously. Your complete exit is given control when the request completes.

MODE=ASYNCTOKEN
The request processes asynchronously. An asynchronous request token is returned in the area specified by REQTOKEN upon invocation of the request. Use the returned request token on the IXLFCOMP macro to determine whether your request has completed.

Note: ANSAREA is a required parameter when MODE=ASYNCTOKEN is specified.

MODE=ASYNCNORESPONSE
The request processes asynchronously. No notification of request completion is provided. The answer area (ANSAREA) fields will not contain valid information when this mode is specified.

Note: You cannot code MODE=ASYNCNORESPONSE with LOCKINDEX, BUFFER, or BUFLIST.

,NEWAUTH=*newauth*
Use this input parameter to specify a new list authority value to replace the existing list authority value of the list specified by LISTNUM.

If NEWAUTH is not specified, the list authority of the LISTNUM list will remain unchanged.

Note: At the time the structure was allocated, the list authority value for each list in the structure was initialized to binary zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the new list authority value.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLIST Version Support" on page 668 for a descriptions of the options available with PLISTVER.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*
Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*
Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:
- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*
Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*
Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,SETCURSOR=HEAD
,SETCURSOR=TAIL
Use this input parameter to indicate that the list cursor and the list cursor direction are to be set.

Note: At the time the structure was allocated, the list cursor for each list in the structure was initialized to binary zeros. The default list cursor direction for all lists in the structure was set to a head-to-tail direction. Use this parameter only with a coupling facility of CFLEVEL=1 or higher.

SETCURSOR=HEAD
The list cursor is to be set to the entry identifier for the first entry on the list and the list cursor direction is to be set in a head-to-tail direction. If the list is empty, the list cursor is reset to binary zeros.

SETCURSOR=TAIL
The list cursor is to be set to the entry identifier for the last entry on the list and the list cursor direction is to be set in a tail-to-head direction. If the list is empty, the list cursor is reset to binary zeros.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If applicable, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 66. Return and Reason Codes for IXLLIST REQUEST=WRITE_LCONTROLS Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.

Table 66. Return and Reason Codes for IXLLIST REQUEST=WRITE_LCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.

Table 66. Return and Reason Codes for IXLLIST REQUEST=WRITE_LCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLIST while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLIST while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLIST in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLIST macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 66. Return and Reason Codes for IXLLIST REQUEST=WRITE_LCONTROLS Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority specified for AUTHCOMP failed the comparison specified by AUTHCOMPTYPE for the list authority of the specified list. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLIST is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.

Table 66. Return and Reason Codes for IXLLIST REQUEST=WRITE_LCONTROLS Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLOCK Services

Description

MVS allows a connected coupling facility application to request the following lock services through the IXLLOCK macro:

- **Obtain** either shared or exclusive ownership of a resource that is not currently owned or is pending ownership by this connected user and optionally assign additional user-defined ownership attributes. (REQUEST=OBTAIN)
- **Alter** the attributes of an owned resource or replace a previous OBTAIN or ALTER request which is pending on the contention exit resource request queue with a more current request. (REQUEST=ALTER)
- **Release** ownership of a resource or, if an outstanding request has been left pending on the contention exit resource request queue, replace the pending request with the more current request to release the resource ownership. As with the ALTER request, you can request that a resource be released before you have received the results of a previous request to OBTAIN or ALTER. (REQUEST=RELEASE)
- **Process multiple** requests for resources with a single IXLLOCK invocation. A lock request block specifies each individual request. You can request that the system service up to 128 requests. The set of request types (OBTAIN, ALTER, or RELEASE) supported by a PROCESSMULT request is a function of the version of the IXLLOCK macro. Version 1 of the IXLLOCK macro supports the PROCESSMULT request type and requires a coupling facility of CFLEVEL=2 or higher. (REQUEST=PROCESSMULT)

The IXLLOCK macro requires that you provide a predefined set of user exit routines that are necessary to accomplish the locking function. You specify the addresses of these routines when you connect to the lock structure with the IXLCONN macro.

- The **contention exit** is executed to resolve contention for a resource. Through the contention exit parameter list (IXLYCEPL), the contention exit can potentially direct XES to grant or deny a request for a resource, modify the ownership characteristics of one or more of the current resource owners, notify a current resource owner that contention exists, or take no action at all. The contention exit that is executed is that of the connected user that XES has assigned contention management responsibilities.
- The **notify exit** can be executed to inform a user that contention exists for a resource that the user owns. Only the contention exit of the connected user chosen to manage the resource contention can request that the notify exit be executed. The notify exit parameter list (IXLYNEPL) provides the user with information about all the current owners and requestors of the resource, in which the user can choose to update its interest in the resource and possibly eliminate the contention. The synchronous update is accomplished with the IXLSYNCH macro.
- The **complete exit** is the means by which XES presents the results of a user's request that could not be processed immediately but was processed asynchronously. The complete exit also is used to notify a user that its ownership state of a resource has been changed (regranted) by the connected user that is managing contention for the resource.
- The **event exit** is the means by which XES reports error and status conditions to connected users. Though not specific to locking functions, use of the event exit must be considered when designing a locking protocol.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued.

AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space See “Restrictions and Limitations”.

Programming Requirements

Before issuing an IXLLOCK service, the connected user must identify itself to the system through the IXLCONN service. On successful completion of the IXLCONN service, a sysplex-wide unique CONNECT token is returned to the user. This token identifies the user’s connection to the lock structure. The connect token must be specified on every IXLLOCK request to ensure that the requesting user is allowed access to the designated lock structure.

The IXLYCON macro provides a list of constants for users of IXLLOCK. Include that macro in your program.

Restrictions and Limitations

The following restrictions apply:

- The caller must provide a 144-byte savearea that starts on a word boundary and is addressable in the caller’s primary address space.
- If the caller is running in access register (AR) ASC mode and specifies a macro parameter using explicit register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage area specified by the REQBUFFER keyword must reside in fixed or disabled reference storage and must be addressable from the caller’s primary address space.
- The PROCESSMULT request type is valid only for structures allocated in a coupling facility of CFLEVEL=2 or higher.

Input Register Information

Before issuing the IXLLOCK macro, the caller must place in GPR 13 the address of a 144-byte save area to be used by XES. With the exception of GPR 13, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLLOCK macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code if GPR15 return code is nonzero
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller of the IXLLOCK macro, the ARs contain:

Register	Contents
0-1	Used as a work register by the system
2-13	Unchanged
14-15	Used as a work register by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro, and restore them after the system returns control.

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLLOCK Version Support

The IXLLOCK macro supports versions in the range of 0 - 2.

- Keywords not specifically noted here are supported by all versions starting with version 0 of the IXLLOCK macro.
- The following keywords and functions are supported by all versions starting with version 1 of the IXLLOCK macro.

MODEVAL	REQUEST=PROCESSMULT
REQBUFFER	MODE=SYNCFAIL
REQNUM	MODE=VALUE
REQPROC	

- The following keyword is supported by all versions starting with version 2 or higher of the IXLLOCK macro.

RNAMELEN

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLLOCK macro is as follows:

main diagram

```
►►──IXLLOCK──b──CONTOKEN=contoken──,REQUEST=──┬─OBTAIN──┤ parameters-1 ├──────┬──┬─────────────────────┬──►
                                              ├─ALTER──┤ parameters-4 ├───────┤  └─,RETCODE=retcode─┘
                                              ├─RELEASE──┤ parameters-6 ├─────┤
                                              └─PROCESSMULT──┤ parameters-8 ├─┘

►──┬──────────────────────┬──┬─,PLISTVER=IMPLIED_VERSION─┬──┬─,MF=S───────────────────────────────────┬──►◄
   └─,RSNCODE=rsncode─────┘  ├─,PLISTVER=MAX─────────────┤  ├─,MF=(L─,mfctrl─┬─,0D──────┬─)──────────┤
                             └─,PLISTVER=plistver────────┘  │                └─,mfattr──┘            │
                                                            └─,MF=(E─,mfctrl─┬─,COMPLETE─┬─)─────────┘
                                                                             └─,COMPLETE─┘
```

parameters-1

```
►►──,RNAME=rname──┬─,RNAMELEN=NO_RNAMELEN─┬──,HASHVAL=hashval──┬─,LOCKDATA=ALL_ZEROES─┬──────────►
                  └─,RNAMELEN=rnamelen────┘                    └─,LOCKDATA=lockdata───┘
```

IXLLOCK Macro

```
   ┌─,STATE=SHR──────────────────────────────┐  ┌─,UDATAVAL=ALL_ZEROES─┐
►──┼─────────────────────────────────────────┼──┼──────────────────────┼──────────────────────────────────►
   ├─,STATE=EXCL─────────────────────────────┤  └─,UDATAVAL=udataval───┘
   └─,STATE=VALUE──,STATEVAL=stateval────────┘

   ┌─,RDATA=NORDATA─────────────────────────┐  ┌─,MODE=SYNCSUSPEND────────────────┐
►──┼────────────────────────────────────────┼──┼──────────────────────────────────┼───────────────────►
   ├─,RDATA=WRITE─┤ parameters-2 ├──────────┤  ├─,MODE=SYNCEXIT───────────────────┤
   └─,RDATA=REACQUIRE─┤ parameters-3 ├──────┘  ├─,MODE=SYNCFAIL───────────────────┤
                                               └─,MODE=VALUE──,MODEVAL=modeval────┘

   ┌─,STARTCONT=DELAYAVG──┐
►──┼──────────────────────┼────────────────────────────────────────────────────────────────────────►◄
   └─,STARTCONT=IMMEDIATE─┘
```

parameters-2

```
►►──,RDATAVAL=rdataval──┬──────────────────┬──┬────────────────────────┬───────────────────────►◄
                        └─,ENTRYID=entryid─┘  └─,ENTRYCOUNT=entrycount─┘
```

parameters-3

```
                        ┌─,CONID=NO_CONID─┐  ┌─,UPDATERDATA=NO─────────────────────────┐
►►──,ENTRYID=entryid────┼─────────────────┼──┼─────────────────────────────────────────┼──────────►◄
                        └─,CONID=conid────┘  └─,UPDATERDATA=YES──,RDATAVAL=rdataval────┘
```

parameters-4

```
                      ┌─,RNAMELEN=NO_RNAMELEN─┐
►►──,RNAME=rname──────┼───────────────────────┼──,HASHVAL=hashval──────────────────────────────────►
                      └─,RNAMELEN=rnamelen────┘

   ┌─,STATE=SHR──────────────────────────────┐  ┌─,UDATAVAL=ALL_ZEROES─┐
►──┼─────────────────────────────────────────┼──┼──────────────────────┼──────────────────────────────────►
   ├─,STATE=EXCL─────────────────────────────┤  └─,UDATAVAL=udataval───┘
   └─,STATE=VALUE──,STATEVAL=stateval────────┘

   ┌─,RDATA=UNCHANGED───────────────────┐  ┌─,MODE=SYNCSUSPEND────────────────┐
►──┼────────────────────────────────────┼──┼──────────────────────────────────┼──────────────────────►◄
   ├─,RDATA=DELETE──────────────────────┤  ├─,MODE=SYNCEXIT───────────────────┤
   └─,RDATA=WRITE─┤ parameters-5 ├──────┘  ├─,MODE=SYNCFAIL───────────────────┤
                                           └─,MODE=VALUE──,MODEVAL=modeval────┘
```

parameters-5

```
►►──,RDATAVAL=rdataval──,ENTRYID=entryid──┬────────────────────────┬───────────────────────────►◄
                                          └─,ENTRYCOUNT=entrycount─┘
```

parameters-6

```
                      ┌─,RNAMELEN=NO_RNAMELEN─┐                      ┌─,UDATAVAL=ALL_ZEROES─┐
►►──,RNAME=rname──────┼───────────────────────┼──,HASHVAL=hashval────┼──────────────────────┼──────►
                      └─,RNAMELEN=rnamelen────┘                      └─,UDATAVAL=udataval───┘
```

```
   ┌─,RDATA=DELETE──────────────────────┐  ┌─,MODE=SYNCSUSPEND──────────────────────┐
►──┼────────────────────────────────────┼──┼────────────────────────────────────────┼──────────────────────────►◄
   └─,RDATA=KEEP──┤ parameters-7 ├──────┘  ├─,MODE=SYNCEXIT─────────────────────────┤
                                           ├─,MODE=NORESPONSE───────────────────────┤
                                           └─,MODE=VALUE──,MODEVAL=modeval──────────┘
```

parameters-7

```
     ┌─,UPDATERDATA=NO───────────────────────────┐
►►───┼───────────────────────────────────────────┼──────────────────────────────────►◄
     └─,UPDATERDATA=YES──,RDATAVAL=rdataval──────┘
```

parameters-8

```
►►──,REQBUFFER=reqbuffer──,REQNUM=reqnum──┬──────────────────────┬─────────────────►◄
                                          └─,REQPROC=reqproc─────┘
```

Parameter Descriptions

The parameter descriptions common to all IXLLOCK request types are listed in alphabetical order. Default values are underlined.

CONTOKEN=*contoken*

Use this input parameter to specify the CONNECT token that was returned in the answer area by the IXLCONN service. CONTOKEN uniquely identifies the user's connection to a lock structure.

To Code: Specify the RS-name or address (using a register from 2 to 12) of a 16-character input field that contains the CONNECT token returned in the answer area by the IXLCONN service.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLOCK Version Support" on page 1001 for a description of the options available with PLISTVER.

,REQUEST=OBTAIN
,REQUEST=ALTER
,REQUEST=RELEASE
,REQUEST=PROCESSMULT
Use this input parameter to specify the type of operation requested.

OBTAIN
Use this input parameter to specify that the connected user is requesting to obtain ownership of the resource identified by the input resource name/hash value pair.

ALTER
Use this input parameter to request a change to one or more of the attributes of a resource that it currently owns. The ALTER option also may be used to replace a previous OBTAIN or ALTER request that is currently pending on the contention exit resource request queue with a more current request.

RELEASE
Use this input parameter to specify that the connected user is requesting to release ownership of the resource.

PROCESSMULT
Use this input parameter to specify that the connected user is requesting that multiple resource requests are to be processed. Each request is specified in a lock request block that the user builds in a storage area.

This request type is valid only for a structure allocated in a coupling facility of CFLEVEL=2 or higher.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

REQUEST=OBTAIN

The parameter descriptions for REQUEST=OBTAIN are listed in alphabetical order. Default values are underlined.

,CONID=NO_CONID
,CONID=*conid*

Use this input parameter to specify the connection identifier that indicates the connection from which the record data entry is being reacquired. If the record data entry designated by ENTRYID is not associated with the connection specified by CONID, the IXLLOCK request will fail. When a record data entry is successfully reacquired, it will become associated with the reacquiring connected user.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains the connection identifier associated with the record data entry to be reacquired.

,ENTRYCOUNT=*entrycount*

Use this output parameter to contain the number of record data entries in the structure that are currently in-use upon successful, synchronous completion of the request. This value is analogous to the IXLYAMDSTRL_LSEC field that is returned by the XES Accounting and Measurement service, IXLMG. The value returned in the ENTRYCOUNT field can be used in conjunction with the value indicating the maximum number of record data entries supported by the allocated lock structure to monitor structure capacity and anticipate “structure full” conditions. The maximum number of record data entries allowed is returned in the CONALOCKMAXRECORDELEMENTS field of the Connect answer area. You can also use the IXLMG macro to retrieve the value, which will be returned in the IXLYAMDSTRL_LSEC field of the IXLMG answer area.

Note that if the request is unsuccessful or being processed asynchronously (in which case the results are presented to the user’s complete exit), the contents of ENTRYCOUNT are not valid.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword output field to contain the number of record data entries in the structure that are currently in-use.

,ENTRYID=*entryid*

Use this input/output parameter to specify the unique identifier assigned to the record data entry.

- When RDATA=WRITE, ENTRYID is an output parameter to contain the unique identifier assigned to the record data entry upon successful, synchronous request completion. If the request is unsuccessful or being processed asynchronously (in which case the results are presented to the user’s complete exit), the contents of ENTRYID are not valid.
- When RDATA=REACQUIRE, ENTRYID is an input parameter to contain the unique identifier of the record data entry to be reacquired. The entry with this identifier must already exist.

ENTRYID is a required keyword when RDATA=REACQUIRE or RDATA=WRITE is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character input/output field to contain the unique identifier assigned to the record data entry.

,HASHVAL=*hashval*

Use this input parameter to specify a hash value associated with the resource name. The hash value along with the resource name serves to fully qualify an IXLLOCK resource. The method of producing the hash value is completely at the discretion of the connected user. Typically, the value provided for this keyword is the output of a user-defined hashing algorithm that receives a resource name as input.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an input fullword field that contains a hash value associated with the resource name.

,LOCKDATA=ALL_ZEROES
,LOCKDATA=*lockdata*

Use this input parameter to specify user-defined data to be associated with this resource. The contents of LOCKDATA are at the discretion of the user and have no meaning to the system. The associated LOCKDATA is presented to this connected user’s complete exit if the OBTAIN request is processed asynchronously. The LOCKDATA is also presented to the complete and notify exits to inform the user of subsequent updates (such as the completion of requests to alter this resource) and status regarding the owned resource.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an eight-character input field that contains connected user-defined data to be associated with this resource.

,MODE=SYNCSUSPEND
,MODE=SYNCEXIT
,MODE=SYNCFAIL
,MODE=VALUE
Use this input parameter to specify how the request should be processed if it is not able to be serviced immediately. If the request is able to be processed immediately, the MODE keyword is ignored and control is returned to the caller with all information regarding the completed request.

SYNCSUSPEND
Indicates that the OBTAIN request is to be handled synchronously. The caller receives control back only when the request is complete. If necessary, the caller is suspended until the request completes.

SYNCEXIT
Indicates that the OBTAIN request is to be handled asynchronously. Return and reason codes are returned to the caller indicating that the request will be processed in this manner. Request completion is reported through the connected user's complete exit. The user-specified complete exit may be given control before control returns to the next sequential instruction after the connected user's IXLLOCK request.

SYNCFAIL
Indicates that the OBTAIN request is to be cancelled if it cannot be processed without delay. Return and reason codes are returned to the caller indicating that the request has been cancelled.

VALUE
Indicates that the contents of MODEVAL are to be used to specify how the request is to be processed if it cannot be serviced immediately.

,MODEVAL=*modeval*
Use this input parameter to specify the value, as represented in IXLYCON, of the desired mode in which the request is to be processed.

The valid IXLYCON constants for an OBTAIN request are:
- IXLMODESYNCSUSPEND
- IXLMODESYNCEXIT
- IXLMODESYNCFAIL

If you specify a value other than one of the IXLYCON constants that is valid for an OBTAIN request, the IXLLOCK request fails with reason code IXLRSNCODEBADMODEVAL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains a value indicating the desired mode in which the request is to be processed.

,RDATA=NORDATA
,RDATA=WRITE
,RDATA=REACQUIRE
Use this input parameter to specify the record data operation, if any, that is to be performed as part of obtaining the specified resource.

NORDATA
Indicates that no record data entry is to be allocated and associated with the specified resource.

WRITE
Indicates that a record data entry is to be allocated and to be associated with the owned resource.

REACQUIRE
Indicates that a record data entry identified by ENTRYID is to be reacquired.

The use of the REACQUIRE option is intended to aid in recovery of resources during recovery scenarios. For example,

- Upon reconnecting, a previously failed-persistent connected user of locking services can re-obtain resources that were held by its previous instance and reacquire the existing record data entries to be associated with the new instances of ownership. The user could potentially use the UPDATERDATA suboption to update the contents of the reacquired record data entries to reflect updated state information.
- A connected user of locking services fails such that the related surviving users wish to recover the resources held by the failing user. The survivors may wish to obtain the specified resources while reacquiring the associated record data entries from the failed connector. The surviving connectors could potentially exploit the CONID suboption to coordinate their processing.

,RDATAVAL=*rdataval*

Use this input/output parameter to specify the user-defined data to be written to the record data entry.

RDATAVAL is a required parameter when RDATA=WRITE is specified. The entry identifier of the record data entry is returned to the connected user in the ENTRYID field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be written to the record data entry. The contents of RDATAVAL are at the discretion of the connected user and have no meaning to the system. The contention exit may modify the value that is to be written to the record data entry as part of granting the request.

If completion of the OBTAIN request is presented to the user synchronously, the input variable will contain the resultant record data value.

,RNAME=*rname*

Use this input parameter to specify the resource name for which the request is to be processed. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the character input field that contains the resource name for which the request is being processed.

,RNAMELEN=NO_RNAMELEN
,RNAMELEN=*rnamelen*

Use this input parameter to specify the length of the resource name identified by RNAME. The resource name length attribute is established on the IXLCONN invocation. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

RNAMELEN is valid only when variable-length resource names are in effect for the lock structure. The value you specify for RNAMELEN must be between 1 and 300 inclusive.

- If you specify RNAMELEN for a lock structure that does not have the variable-length name attribute, the system rejects the request with reason code IXLRSNCODENOVARRNAME.
- If you specify a value for RNAMELEN that is different from the actual length of RNAME, the system uses the RNAMELEN value as the length of RNAME.
- If you do not specify RNAMELEN, the length of the resource name defaults to 64 bytes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the resource name length of the resource identified by RNAME.

,STATE=SHR
,STATE=EXCL
,STATE=VALUE

Use this input parameter to specify the state in which the connected user is requesting to own the resource.

Note that the OBTAIN request may be granted by the contention exit with a STATE different from what was requested. If a user's protocol is exploiting the capability for the contention exit to grant requests with a STATE different than requested, it is recommended that STATE=VALUE be specified for this option to ensure that an output variable will be available to contain the resultant state when request completion is reported synchronously.

SHR
Specifies a shared state.

EXCL
Specifies an exclusive state.

VALUE
Specifies that the contents of STATEVAL are to be used to identify the ownership state.

,STATEVAL=*stateval*
Use this input/output parameter to specify the value, as represented in IXLYCON, of the desired ownership state.

Note that if a value other than the IXLYCON constants for shared or exclusive is specified, the resource will be requested in the default STATE of share.

Upon successful, synchronous request completion, the input variable will contain the state in which ownership of the resource was granted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains a value indicating the desired ownership state.

,UDATAVAL=ALL_ZEROES
,UDATAVAL=*udataval*
Use this input/output parameter to specify user-defined data to be associated with the resource. The contents of UDATAVAL are at the discretion of the connected user and have no meaning to the system. UDATAVAL is presented to the complete, notify, and contention exits when driven.

Note that the contents of UDATAVAL may be modified by the contention exit as part of granting or denying the request. The contents of UDATAVAL have no meaning to the system.

If request completion is reported to the connected user synchronously, UDATAVAL, if specified, will contain the resultant user data value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be associated with the resource.

,UPDATERDATA=NO
,UPDATERDATA=YES
Use this input parameter to specify whether or not to update the contents of the record data entry at the time it is reacquired.

NO
Indicates that the data in the record data entry is not to be updated.

YES
Indicates that the data in the record data entry is to be updated with the data specified by RDATAVAL.

RDATAVAL is a required keyword when UPDATERDATA=YES is specified.

REQUEST=ALTER

The parameter descriptions for REQUEST=ALTER are listed in alphabetical order. Default values are underlined.

,ENTRYCOUNT=*entrycount*
Use this output parameter to contain the number of record data entries in the structure that are currently in-use upon successful, synchronous completion of the request. This value is analogous to the IXLYAMDSTRL_LSEC field that is returned by the XES Accounting and Measurement service, IXLMG. The value returned in the ENTRYCOUNT field can be used in conjunction with the value

indicating the maximum number of record data entries supported by the allocated lock structure to monitor structure capacity and anticipate “structure full” conditions. The maximum number of record data entries allowed is returned in the CONALOCKMAXRECORDELEMENTS field of the Connect answer area. You can also use the IXLMG macro to retrieve the value, which will be returned in the IXLYAMDSTRL_LSEC field of the IXLMG answer area.

Note that if the request is unsuccessful or being processed asynchronously (in which case the results are presented to the user’s complete exit), the contents of ENTRYCOUNT are not valid.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword output field to contain the number of record data entries in the structure that are currently in-use.

,ENTRYID=*entryid*

Use this output parameter to specify the unique identifier assigned to the record data entry upon successful, synchronous request completion. If a new record data entry was allocated by this request, the identifier indicates the newly allocated entry. If the request resulted in the update of an existing record data entry that had been associated with this resource through a previous OBTAIN or ALTER request, the identifier indicates the updated entry. If the request is unsuccessful or being processed asynchronously (in which case the results are presented to the user’s complete exit), the contents of ENTRYID are not valid.

ENTRYID is a required keyword when RDATA=WRITE is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character output field to contain the unique identifier assigned to the record data entry.

,HASHVAL=*hashval*

Use this input parameter to specify a hash value associated with the resource name. The hash value along with the resource name serves to fully qualify an IXLLOCK resource. The method of producing the hash value is completely at the discretion of the connected user. Typically, the value provided for this keyword is the output of a user-defined hashing algorithm that receives a resource name as input.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an input fullword field that contains a hash value associated with the resource name.

,MODE=SYNCSUSPEND
,MODE=SYNCEXIT
,MODE=SYNCFAIL
,MODE=VALUE

Use this input parameter to specify how the request should be processed if it is not able to be serviced immediately. If the request is able to be processed immediately, the MODE keyword is ignored and control is returned to the caller with all information regarding the completed request.

SYNCSUSPEND
Indicates that the ALTER request is to be handled synchronously. The caller receives control back only when the request is complete. If necessary, the caller is suspended until the request completes.

SYNCEXIT
Indicates that the ALTER request is to be handled asynchronously. Return and reason codes are returned to the caller indicating that the request will be processed in this manner. Request completion is reported through the connected user’s complete exit. The user-specified complete exit may be given control before control returns to the next sequential instruction after the connected user’s IXLLOCK request.

SYNCFAIL
Indicates that the ALTER request is to be cancelled if it cannot be processed without delay. Return and reason codes are returned to the caller indicating that the request has been cancelled.

VALUE
Indicates that the contents of MODEVAL are to be used to specify how the request is to be processed if it cannot be serviced immediately.

,MODEVAL=*modeval*

Use this input parameter to specify the value, as represented in IXLYCON, of the desired mode in which the request is to be processed.

The valid IXLYCON constants for an ALTER request are:

- IXLMODESYNCSUSPEND
- IXLMODESYNCEXIT
- IXLMODESYNCFAIL

If you specify a value other than one of the IXLYCON constants that is valid for an ALTER request, the IXLLOCK request fails with reason code IXLRSNCODEBADMODEVAL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains a value indicating the desired mode in which the request is to be processed.

,RDATA=UNCHANGED
,RDATA=DELETE
,RDATA=WRITE

Use this input parameter to specify the record data operation, if any, that is to be performed as part of obtaining the specified resource.

Note that specification of the WRITE or DELETE options will result in a parameter error if the lock structure indicated by the input CONTOKEN does not provide recording capabilities.

UNCHANGED
Indicates that the record data entry associated with this resource, if any, is not to be changed.

DELETE
Indicates that the record data entry associated with this resource is to be deleted. If no record data entry is currently allocated and associated with this resource, then this keyword is ignored.

WRITE
Indicates that the record data entry identified by ENTRYID is to be updated with the data specified by the RDATAVAL keyword. If a record data entry is currently associated with the resource, its contents will be updated. Otherwise, a new entry will be allocated.

,RDATAVAL=*rdataval*

Use this input/output parameter to specify the user-defined data to be written to the record data entry.

RDATAVAL is a required parameter when RDATA=WRITE is specified. The entry identifier of the record data entry is returned to the connected user in the ENTRYID field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be written to the record data entry. The contents of RDATAVAL are at the discretion of the connected user and have no meaning to the system. The contention exit may modify the value that is to be written to the record data entry as part of granting the request.

If completion of the OBTAIN request is presented to the user synchronously, the input variable will contain the resultant record data value.

,RNAME=*rname*

Use this input parameter to specify the resource name for which the request is to be processed. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the character input field that contains the resource name for which the request is being processed.

,RNAMELEN=NO_RNAMELEN
,RNAMELEN=*rnamelen*

Use this input parameter to specify the length of the resource name identified by RNAME. The resource name length attribute is established on the IXLCONN invocation. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

RNAMELEN is valid only when variable-length resource names are in effect for the lock structure. The value you specify for RNAMELEN must be between 1 and 300 inclusive.

- If you specify RNAMELEN for a lock structure that does not have the variable-length name attribute, the system rejects the request with reason code IXLRSNCODENOVARRNAME.
- If you specify a value for RNAMELEN that is different from the actual length of RNAME, the system uses the RNAMELEN value as the length of RNAME.
- If you do not specify RNAMELEN, the length of the resource name defaults to 64 bytes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the resource name length of the resource identified by RNAME.

,STATE=SHR
,STATE=EXCL
,STATE=VALUE

Use this input parameter to specify the state in which the connected user is requesting to own the resource.

Note that the ALTER request may be granted by the contention exit with a STATE different from what was requested. If a user’s protocol is exploiting the capability for the contention exit to grant requests with a STATE different than requested, it is recommended that STATE=VALUE be specified for this option to ensure that an output variable will be available to contain the resultant state when request completion is reported synchronously.

SHR
Specifies a shared state.

EXCL
Specifies an exclusive state.

VALUE
Specifies that the contents of STATEVAL are to be used to identify the ownership state.

,STATEVAL=*stateval*

Use this input parameter to specify the value, as represented in IXLYCON, of the desired ownership state.

Note that if a value other than the IXLYCON constants for shared or exclusive is specified, the resource will be requested in the default STATE of share.

Upon successful, synchronous request completion, the input variable will contain the state in which ownership of the resource was granted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains a value indicating the desired ownership state.

,UDATAVAL=ALL_ZEROES
,UDATAVAL=*udataval*

Use this input/output parameter to specify user-defined data to be associated with the resource. The contents of UDATAVAL are at the discretion of the connected user and have no meaning to the system. UDATAVAL is presented to the complete, notify, and contention exits when driven.

Note that the contents of UDATAVAL may be modified by the contention exit as part of granting or denying the request. The contents of UDATAVAL have no meaning to the system.

If request completion is reported to the connected user synchronously, UDATAVAL, if specified, will contain the resultant user data value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be associated with the resource.

REQUEST=RELEASE

The parameter descriptions for REQUEST=RELEASE are listed in alphabetical order. Default values are underlined.

,HASHVAL=*hashval*

Use this input parameter to specify a hash value associated with the resource name. The hash value along with the resource name serves to fully qualify an IXLLOCK resource. The method of producing the hash value is completely at the discretion of the connected user. Typically, the value provided for this keyword is the output of a user-defined hashing algorithm that receives a resource name as input.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an input fullword field that contains a hash value associated with the resource name.

,MODE=SYNCSUSPEND
,MODE=SYNCEXIT
,MODE=NORESPONSE
,MODE=VALUE

Use this input parameter to specify how the request should be processed if it is not able to be serviced immediately. If the request is able to be processed immediately, the MODE keyword is ignored and control is returned to the caller with all information regarding the completed request.

SYNCSUSPEND

Indicates that the RELEASE request be handled synchronously. The caller receives control back only when the request is complete. If necessary, the caller is suspended until the request completes.

SYNCEXIT

Indicates that the RELEASE request be handled asynchronously. Return and reason codes are returned to the caller indicating that the request will be processed in this manner. Request completion is reported through the connected user’s complete exit. The user-specified complete exit may be given control before control returns to the next sequential instruction after the connected user’s IXLLOCK request.

NORESPONSE

Indicates that the requestor does not want to be informed when the RELEASE request is complete. Return and reason codes are returned to the caller indicating that the request will be processed asynchronously. However, the connected user’s complete exit will not be invoked to report request completion.

VALUE

Indicates that the contents of MODEVAL are to be used to specify how the request is to be processed if it cannot be serviced immediately.

,MODEVAL=*modeval*

Use this input parameter to specify the value, as represented in IXLYCON, of the desired mode in which the request is to be processed.

The valid IXLYCON constants for a RELEASE request are:
- IXLMODESYNCSUSPEND
- IXLMODESYNCEXIT
- IXLMODENORESPONSE

If you specify a value other than one of the IXLYCON constants that is valid for a RELEASE request, the IXLLOCK request fails with reason code IXLRSNCODEBADMODEVAL.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte input field that contains a value indicating the desired mode in which the request is to be processed.

,RDATA=DELETE

,RDATA=KEEP

Use this input parameter to specify the record data operation, if any, that is to be performed as part of obtaining the specified resource.

DELETE

Indicates that the record data entry associated with this resource is to be deleted. If no record data entry is currently allocated and associated with this resource, then this keyword is ignored.

KEEP

Indicates that the record data entry associated with this resource is to be kept even though ownership of the resource is being relinquished.

Note that the record data entry that was kept may eventually be re-associated with a new resource through the REACQUIRE option of IXLLOCK REQUEST=OBTAIN or manipulated through the XES record data service, IXLRT. If the KEEP option is specified and no associated record data entry exists, informational return and reason codes will be returned to the invoker.

,RDATAVAL=*rdataval*

Use this input/output parameter to specify the user-defined data to be written to the record data entry. The contents of RDATAVAL are at the discretion of the connected user and have no meaning to the system. The contention exit may modify the value that is to be written to the record data entry as part of granting the request.

If completion of the OBTAIN request is presented to the user synchronously, the input variable will contain the resultant record data value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be written to the record data entry. The contents of RDATAVAL are at the discretion of the connected user and have no meaning to the system. The contention exit may modify the value that is to be written to the record data entry as part of granting the request.

,RNAME=*rname*

Use this input parameter to specify the resource name for which the request is to be processed. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the character input field that contains the resource name for which the request is being processed.

,RNAMELEN=NO_RNAMELEN
,RNAMELEN=*rnamelen*

Use this input parameter to specify the length of the resource name identified by RNAME. The resource name length attribute is established on the IXLCONN invocation. The resource name and length along with the hash value serve to fully qualify an IXLLOCK request.

RNAMELEN is valid only when variable-length resource names are in effect for the lock structure. The value you specify for RNAMELEN must be between 1 and 300 inclusive.

- If you specify RNAMELEN for a lock structure that does not have the variable-length name attribute, the system rejects the request with reason code IXLRSNCODENOVARRNAME.
- If you specify a value for RNAMELEN that is different from the actual length of RNAME, the system uses the RNAMELEN value as the length of RNAME.
- If you do not specify RNAMELEN, the length of the resource name defaults to 64 bytes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input field that contains the resource name length of the resource identified by RNAME.

,UDATAVAL=ALL_ZEROES
,UDATAVAL=*udataval*

Use this input/output parameter to specify user-defined data to be associated with the resource. The contents of UDATAVAL are at the discretion of the connected user and have no meaning to the system. UDATAVAL is presented to the complete, notify, and contention exits when driven.

Note that while a RELEASE request cannot be denied by the contention exit, the contents of the user data associated with the request may be modified.

If request completion is reported to the connected user synchronously, UDATAVAL, if specified, will contain the resultant user data value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field that contains user-defined data to be associated with the resource.

,UPDATERDATA=NO
,UPDATERDATA=YES
Use this input parameter to specify whether or not to update the data in the record data entry that has been kept.

NO
Indicates that the data in the record data entry is not to be updated.

YES
Indicates that the data in the record data entry is to be updated with the data specified by RDATAVAL. RDATAVAL is a required keyword when UPDATERDATA=YES is specified.

REQUEST=PROCESSMULT

This request type is valid only for a structure allocated in a coupling facility of CFLEVEL=2 or higher. The parameter descriptions for REQUEST=PROCESSMULT are listed in alphabetical order.

,REQBUFFER=*reqbuffer*
Use this input parameter to specify the virtual storage area that contains from 1 to 128 lock request blocks that correspond to resource requests. The macro IXLYLRB maps each lock request block. The virtual storage area must reside in fixed or disabled reference storage and be addressable from the caller's primary address space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an input buffer that contains the lock request blocks to be processed. The length of the buffer is determined by the number of lock request blocks that it contains.

,REQNUM=*reqnum*
Use this input parameter to specify the number of lock request blocks that are in the area identified by REQBUFFER. The value of REQNUM must be in the range of 1 to 128.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword input field that contains the number of lock request blocks in the REQBUFFER area.

,REQPROC=*reqproc*
Use this output parameter to contain the number of lock request blocks that the system processed.

- If all lock request blocks were processed, this value is equal to the value specified for the REQNUM keyword.
- If an error occurred so that only a partial set of lock request blocks were processed, this value indicates the number of lock request blocks that were processed before the error occurred. The LRBs that follow the LRB designated by REQPROC are in an indeterminate state.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword output field to contain the number of lock request blocks that the system processed.

ABEND Codes

None.

Return and Reason Codes

When the IXLLOCK macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 67. Return and Reason Codes for the IXLLOCK Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** IXLLOCK request successful. • For an OBTAIN or ALTER request, ownership has been granted in the requested state. User data is the same as requested. • For a RELEASE request specifying MODE=SYNCSUSPEND or MODE=SYNCEXIT, processing is complete. • For a RELEASE request specifying MODE=NORESPONSE, the request has been accepted. • For a PROCESSMULT request, all LRB processing is complete. Individual return and reason codes are returned for each LRB. **Action:** None.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** Request will be processed asynchronously. If MODE=SYNCEXIT was specified, request completion will be reported through the user's complete exit. For IXLLOCK RELEASE requests that specified MODE-NORESPONSE, request completion will not be reported to the connected user. **Action:** If you specified MODE=SYNCEXIT, the user's complete exit will be given control when the request completes.
4	xxxx0418	**Equate Symbol:** IXLRSNCODENOELEMENTTOKEEP **Meaning:** The IXLLOCK RELEASE request is successful. The user specified to keep the record data entry associated with the resource, but there was no data entry to keep. **Action:** None expected. However, if you were expecting a record data entry to be present, determine why it was not.
4	xxxx0419	**Equate Symbol:** IXLRSNCODENOUPDATEONKEEP **Meaning:** The IXLLOCK RELEASE request is successful. The user specified to keep the associated record data entry and update its contents. The element was kept, but its contents were unable to be updated. **Action:** Consider using IXLRT to update the contents of the record data entry.

Table 67. Return and Reason Codes for the IXLLOCK Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx0424	**Equate Symbol:** IXLRSNCODENODELETEONRELEASE **Meaning:** The IXLLOCK RELEASE request is successful. The user specified to delete the associated record data entry, but the entry was unable to be deleted. **Action:** None expected.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** Program error. The version number in the parameter is not valid. This usually indicates that the level of the macro is incompatible with the level of the XES service code. **Action:** Verify that your program was assembled with the correct macro library for the release of MVS that your program is running on.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Caller was not enabled. **Action:** Verify that the program is enabled for I/O and external interrupts.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The input contoken is not valid. The contoken may no longer be valid for one of the following reasons: disconnect has occurred, EOT of the connector's task, input contoken is not the contoken returned from IXLCONN, the request was issued outside the connector's address space, or the contoken has been invalidated for rebuild. **Action:** Verify that the CONTOKEN value specified is valid and for the correct structure.

Table 67. Return and Reason Codes for the IXLLOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0810	**Equate Symbol:** IXLRSNCODERESOURCENOTFOUND **Meaning:** Program error. Resource specified on ALTER or RELEASE request is not owned or pending ownership. Note that this condition can occur when the request to obtain ownership failed or was denied by the contention exit of the connected user who has been selected to manage the resource contention. **Action:** Retry request to gain (OBTAIN) ownership of the resource.
8	xxxx0812	**Equate Symbol:** IXLRSNCODEALREADYOWNED **Meaning:** Program error. Resource specified on OBTAIN request is already owned. **Action:** Check protocol to determine why duplicate request for resource has been issued.
8	xxxx0813	**Equate Symbol:** IXLRSNCODEALREADYPENDING **Meaning:** Program error. Resource specified on OBTAIN request is already pending ownership. **Action:** Check protocol to determine why duplicate request for resource has been issued.
8	xxxx0816	**Equate Symbol:** IXLRSNCODENORTEXISTS **Meaning:** Program error. A request to WRITE or REACQUIRE a record data entry was unable to be processed due to recording not being active in the lock structure indicated by the input contoken. For recording to be active, the first user to connect to the lock structure must have specified RECORD=YES on its IXLCONN invocation. **Action:** In order for recording to become active, that structure would have to be deleted and reallocated with RECORD=YES.
8	xxxx0817	**Equate Symbol:** IXLRSNCODEBADCONID **Meaning:** Program error. The request to conditionally reacquire an existing record data entry based on the connection with which it is associated has failed. The record data entry to be reacquired was not associated with the connection indicated by the input CONID keyword. **Action:** Verify that the CONID value was specified correctly.
8	xxxx0818	**Equate Symbol:** IXLRSNCODENOTLOCKSTR **Meaning:** Program error. The contoken specified does not represent a lock structure. **Action:** Verify that the CONTOKEN is specified correctly and is for the intended structure.
8	xxxx0844	**Equate Symbol:** IXLRSNCODEBADID **Meaning:** Program error. The attempt to REACQUIRE an existing record data entry has failed because the input ENTRYID does not designate an existing record data entry. **Action:** Verify that the ENTRYID is specified correctly.

Table 67. Return and Reason Codes for the IXLLOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0876	**Equate Symbol:** IXLRSNCODEBADREQNUM **Meaning:** Program error. The REQNUM value specified on IXLLOCK REQUEST=PROCESSMULT is not valid. The value must be between 1 and 128 inclusive. Processing is halted with no entries in the REQBUFFER having been processed. **Action:** Verify that the REQNUM value is specified correctly.
8	xxxx0877	**Equate Symbol:** IXLRSNCODEBADLRBTYPE **Meaning:** Program error. A lock request block (LRB) that is input on an IXLLOCK REQUEST=PROCESSMULT request contains a value in the LRB_XTYPE field that is not valid. **Action:** Verify that the LRB_XTYPE field contains the value of LRB_XTYPE_RELEASEVERS0. If specified, the REQPROC field contains the number of LRBs processed before this error was encountered.
8	xxxx0878	**Equate Symbol:** IXLRSNCODEBADREQBUFFER **Meaning:** Program error. An error occurred while XES was attempting to access the storage area defined by REQBUFFER. The number of LRBs processed is returned in the REQPROC field if specified. **Action:** Verify that the virtual storage area defined by REQBUFFER resides in fixed or disabled reference storage and is addressable from the your primary address space.
8	xxxx0879	**Equate Symbol:** IXLRSNCODEBADMODEVAL **Meaning:** Program error. The value specified for MODEVAL is not valid. **Action:** Verify that the value specified for MODEVAL is one of the possible mode value constants provided in the IXLYCON macro and that it is a valid value for the type of request being processed. If an LRB contains a mode value that is not supported, the system halts the request. The number of LRBs processed prior to the error is returned in the REQPROC field.
8	xxxx087A	**Equate Symbol:** IXLRSNCODEBADRNAMELEN **Meaning:** Program error. The value specified for RNAMELEN is not valid. **Action:** Verify that the length specified for RNAMELEN is between 1 and 300.
8	xxxx087B	**Equate Symbol:** IXLRSNCODENOVARRNAME **Meaning:** Program error. An IXLLOCK request that specified a variable length resource name is not valid because the variable length name attribute is not in effect for the lock structure represented by the input contoken. Specify the variable length name attribute at structure allocation time with the RNAMELEN keyword of the IXLCONN macro. **Action:** If you want to use variable-length resource names, ensure that the initial IXLCONN invocation to connect to the lock structure specifies RNAMELEN=VAR300.

Table 67. Return and Reason Codes for the IXLLOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. There is no connectivity to the lock structure. This may occur due to operator commands such as VARY PATH OFFLINE or CONFIG CHP OFFLINE or hardware errors such as coupling facility or path failures. The contoken will be invalidated. **Action:** Either disconnect from the structure using IXLDISC or rebuild the structure using IXLREBLD.
C	xxxx0C0B	**Equate Symbol:** IXLRSNCODERTFULL **Meaning:** Environmental error. Record portion of the lock structure is full. **Action:** If your protocol allows, attempt to rebuild the lock structure or alter its size so that additional record data might be available.
C	xxxx0C0D	**Equate Symbol:** IXLRSNCODESUPERSEDED **Meaning:** Environmental error. OBTAIN or ALTER request has been superseded by a more current request for the resource. **Action:** None expected.
C	xxxx0C0F	**Equate Symbol:** IXLRSNCODEDENIED **Meaning:** Environmental error. OBTAIN or ALTER request is not granted. Resource request is denied (cancelled) by a related connected user that is managing the contention environment for the resource. **Action:** Retry request at a later time.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. Prior to completion of the request, the lock structure failed. **Action:** Attempt to rebuild the structure using IXLREBLD or disconnect from the structure using IXLDISC.

Table 67. Return and Reason Codes for the IXLLOCK Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C4C	**Equate Symbol:** IXLRSNCODERESOURCENOLONGEROWNED **Meaning:** Environmental error. An IXLLOCK ALTER or IXLLOCK RELEASE request for a resource failed because the resource is no longer owned. **Action:** Check your protocol to determine why the resource is no longer owned.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0C69	**Equate Symbol:** IXLRSNCODENODELAY **Meaning:** Environmental error. An IXLLOCK request in which the user specified MODE=SYNCFAIL encountered a delay. The request is cancelled. **Action:** Retry request at a later time.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** Failure in XES processing. The state of the involved structure and the disposition of the request are unpredictable. **Action:** Save the reason code information and contact the IBM support center.

IXLLSTC — XES List Structure Control Services

Description

The IXLLSTC macro provides operations to atomically perform locking functions, monitor list and sublist state changes, read and write list controls, read event queue controls and event monitor controls, and dequeue event monitor controls.

The IXLLSTC macro provides list services equivalent to the following IXLLIST request types:
- IXLLIST REQUEST=DEQ_EVENTQ
- IXLLIST REQUEST=LOCK
- IXLLIST REQUEST=MONITOR_EVENTQ
- IXLLIST REQUEST=MONITOR_LIST
- IXLLIST REQUEST=MONITOR_SUBLIST
- IXLLIST REQUEST=MONITOR_SUBLISTS
- IXLLIST REQUEST=READ_EMCONTROLS
- IXLLIST REQUEST=READ_EQCONTROLS
- IXLLIST REQUEST=READ_LCONTROLS
- IXLLIST REQUEST=WRITE_LCONTROLS

New functions are available with the IXLLSTC macro at OS/390 Release 9. These new functions require that the list structure be allocated in a coupling facility of CFLEVEL=9 or higher.
- Keyrange monitoring (with REQUEST=MONITOR_KEYRANGE) allows you to monitor the empty or not-empty state of a set of keyrange values, based on user-defined threshold counts. Both the threshold counts and the starting and ending keyrange values are defined with IXLLSTC REQUEST=WRITE_LCONTROLS.
- Support for secondary keys is provided on the REQUEST=READ_EQCONTROLS, REQUEST=READ_EMCONTROLS, REQUEST=DEQ_EVENTQ, REQUEST=MONITOR_EVENTQ, MONITOR_SUBLIST, MONITOR_SUBLISTS request types.
- Support for a request that whenever a list entry is queued to a monitored sublist, an EMC is queued to the registered user's event queue is provided on the REQUEST=MONITOR_SUBLIST and REQUEST=MONITOR_SUBLISTS request types.

See *z/OS MVS Programming: Sysplex Services Guide* for information about using the IXLLSTC macro.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The current primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued for the structure.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Disabled callers must be legally disabled by holding the CPU lock and cannot hold other disabled locks. Enabled callers must not hold any locks. When MODE=SYNCSUSPEND is specified, the caller must be enabled.
Control parameters:	See "Restrictions" on page 1022

Programming Requirements

- If your program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLLSTC. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.
- Include the IXLYCON mapping macro in your program. This macro provides a list of equate symbols for users of XES services and exits.
- Include mapping macros IXLYEMC, IXLYLAA, IXLYLCTL, IXLYLMI, and IXLYMSRI in your program as necessary. Table 68 lists these macros, the information and areas they map, and the particular IXLLSTC requests and parameters they apply to.

Table 68. Mapping Macros for IXLLSTC

Mapping Macro	Information	Area	Request/Parameter
IXLYEMC	Event monitor controls	BUFLIST buffers, BUFFER area	DEQ_EVENTQ
IXLYLAA	Answer area output	ANSAREA area	All requests
IXLYLMI	List monitoring information	BUFLIST buffers, BUFFER area	READ_LCONTROLS
IXLYMSRI	Monitor sublists registration information	BUFLIST buffers, BUFFER area	MONITOR_SUBLISTS

Restrictions

- If you specify BUFLIST and PAGEABLE=YES, all of the buffers in the list must be in the same area of storage; you cannot mix common and private storage addresses for the buffers in the list.
- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).
- The caller's parameter list must be addressable in the caller's primary address space.
- If the caller is running in AR ASC mode and specifies a parameter using explicit register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage areas specified by the ADJAREA and ANSAREA parameters must be addressable in the caller's primary address space or from the caller's PASN access list.
- The virtual storage areas specified by parameters other than ADJAREA and ANSAREA can be addressable in the caller's primary, secondary, or home address spaces, from the PASN access list, or from the dispatchable unit access list (DU-AL).
- If the caller is disabled then the parameter list and all storage areas addressed by the macro parameters must reside in either nonpageable or disabled reference storage.

Input Register Information

Before issuing the IXLLSTC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0	Reason code
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register **Contents**

0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLLSTC Version Support

The IXLLSTC macro supports versions 0, 1, 2 and 4 — the version number corresponds to the level of the IXLLIST or IXLLSTC macro in which a keyword or function was introduced. The higher the version number, the more recent the version of the macro. Some IXLLSTC keywords are supported for multiple versions, but have different functions for each version.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLLSTC macro.
- The following keywords are supported by all versions starting with version 1 and higher of the IXLLSTC macro:

LISTKEY	MAXLISTKEY	SETCURSOR

- The following keywords and functions are supported by all versions starting with version 2 and higher of the IXLLSTC macro:

ENDINDEX	MOSVECTOR	STARTINDEX
ENTRYKEY	NOTIFICATION	UNC
KEYTYPE		

- The following keywords and functions are supported by all versions starting with version 4 and higher of the IXLLSTC macro:

KEYRANGEEND	KRNOTEMPTY	LISTNOTEMPTY
KEYRANGESTART	LISTEMPTY	SECONDARYKEY
KREMPTY		

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Summary of Version-Dependent Parameter Functions

Different parameter list versions may be generated depending on the vlaue specified for certain keywords. The following table summarizes the minimum version required to support such specifications. When specifying PLISTVER, be sure that it is at least as high as the highest version number of all the key values being used.

Table 69. IXLLSTC Version Support

Keyword	Version	Key Value
REQUEST	0	LOCK, READ_LCONTROLS,
		WRITE_LCONTROLS, MONITOR_LIST
	2	READ_EQCONTROLS, READ_EMCONTROLS,
		DEQ_EVENTQ, MONITOR_EVENTQ,
		MONITOR_SUBLIST, MONITOR_SUBLISTS
	4	MONITOR_KEYRANGE
KEYTYPE	2	ENTRY

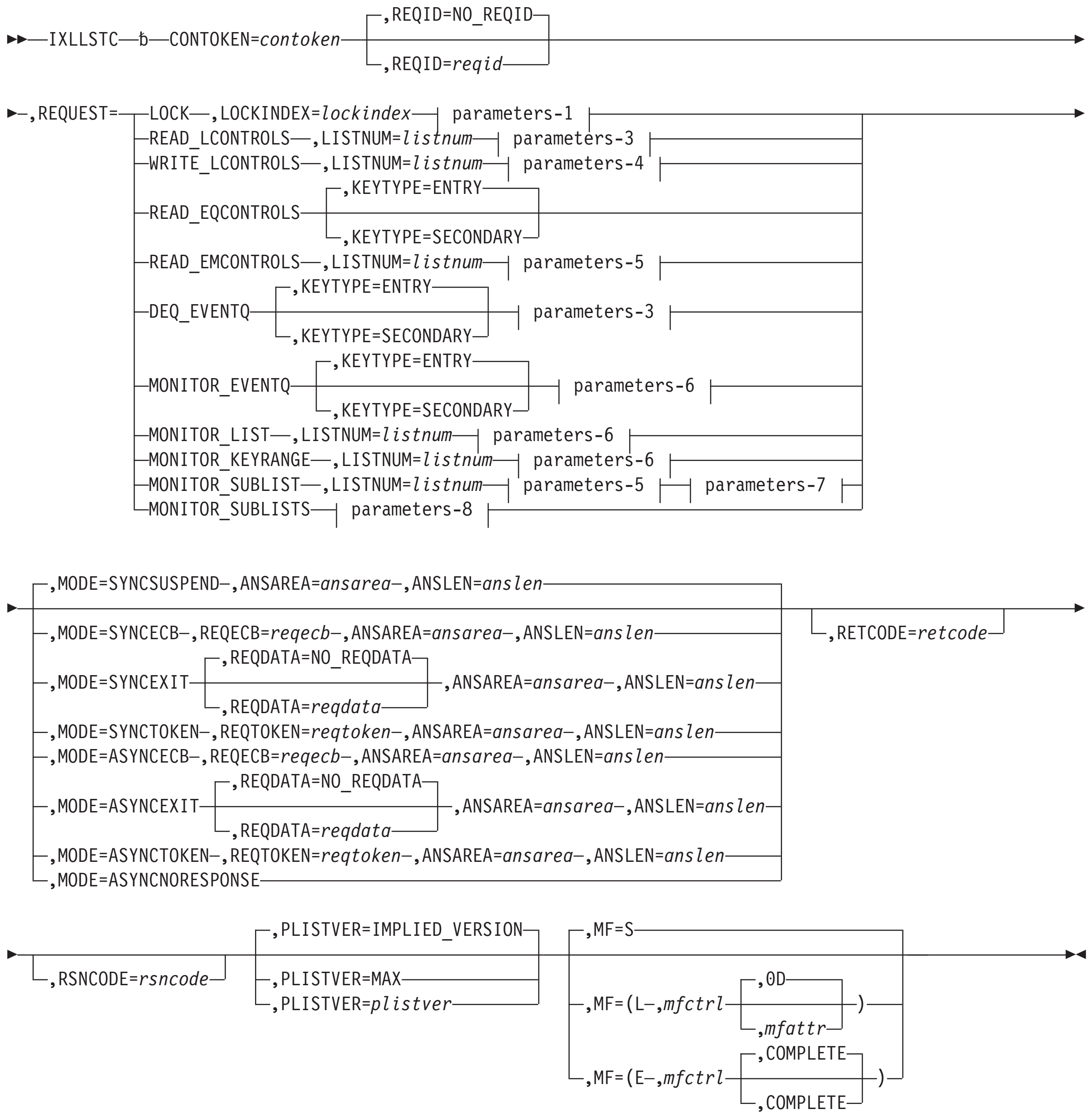
Table 69. IXLLSTC Version Support (continued)

Keyword	Version	Key Value
	4	SECONDARY
NOTIFICATION	2	FIRST
	4	EVERY

Syntax Diagram

The syntax diagram for IXLLSTC is as follows:

main diagram


IXLLSTC
b
CONTOKEN=contoken
,REQID=NO_REQID
,REQID=reqid
,REQUEST=
LOCK ,LOCKINDEX=lockindex parameters-1
READ_LCONTROLS ,LISTNUM=listnum parameters-3
WRITE_LCONTROLS ,LISTNUM=listnum parameters-4
READ_EQCONTROLS ,KEYTYPE=ENTRY ,KEYTYPE=SECONDARY
READ_EMCONTROLS ,LISTNUM=listnum parameters-5
DEQ_EVENTQ ,KEYTYPE=ENTRY ,KEYTYPE=SECONDARY parameters-3
MONITOR_EVENTQ ,KEYTYPE=ENTRY ,KEYTYPE=SECONDARY parameters-6
MONITOR_LIST ,LISTNUM=listnum parameters-6
MONITOR_KEYRANGE ,LISTNUM=listnum parameters-6
MONITOR_SUBLIST ,LISTNUM=listnum parameters-5 parameters-7
MONITOR_SUBLISTS parameters-8
,MODE=SYNCSUSPEND ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=SYNCECB ,REQECB=reqecb ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=SYNCEXIT ,REQDATA=NO_REQDATA ,REQDATA=reqdata ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=SYNCTOKEN ,REQTOKEN=reqtoken ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=ASYNCECB ,REQECB=reqecb ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=ASYNCEXIT ,REQDATA=NO_REQDATA ,REQDATA=reqdata ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=ASYNCTOKEN ,REQTOKEN=reqtoken ,ANSAREA=ansarea ,ANSLEN=anslen
,MODE=ASYNCNORESPONSE
,RETCODE=retcode
,RSNCODE=rsncode
,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=plistver
,MF=S
,MF=(L ,mfctrl ,0D ,mfattr)
,MF=(E ,mfctrl ,COMPLETE ,COMPLETE)

parameters-1

```
►►──,LOCKOPER=──┬─SET─┤ parameters-2 ├──────────────────┬──────────────────────►◄
                ├─RESET─┬─,LOCKCOMP=NO_LOCKCOMP─┬───────┤
                │       └─,LOCKCOMP=lockcomp────┘       │
                ├─TEST─┬─,LOCKCOMP=NO_LOCKCOMP─┬────────┤
                │      └─,LOCKCOMP=lockcomp────┘        │
                └─READNEXT─┬─,LOCKCOMP=NO_LOCKCOMP─┬────┘
                           └─,LOCKCOMP=lockcomp────┘
```

parameters-2

```
►►──┬─,LOCKMODE=UNCOND──────────────────────────────┬──┬─,LOCKDATA=NO_LOCKDATA─┬──►◄
    └─,LOCKMODE=COND─┬─,LOCKCOMP=NO_LOCKCOMP─┬──────┘  └─,LOCKDATA=lockdata────┘
                     └─,LOCKCOMP=lockcomp────┘
```

parameters-3

```
►►──┬─,BUFLIST=buflist─┤ parameters-9 ├──┬──────────────────────────────────►◄
    └─,BUFFER=buffer─┤ parameters-10 ├───┘
```

parameters-4

```
►►──,AUTHCOMP=authcomp──┬─,NEWAUTH=NO_NEWAUTH─┬──┬─,LISTLIMIT=NO_LISTLIMIT─┬──────►
                        └─,NEWAUTH=newauth────┘  └─,LISTLIMIT=listlimit────┘

►──┬─,LISTDESC=NO_LISTDESC─┬──┬─,LISTKEY=NO_LISTKEY─┬──┬─,MAXLISTKEY=NO_MAXLISTKEY─┬──►
   └─,LISTDESC=listdesc────┘  └─,LISTKEY=listkey────┘  └─,MAXLISTKEY=maxlistkey────┘

►──┬─,SETCURSOR=NO_SETCURSOR─┬─────────────────────────────────────────────────────►
   ├─,SETCURSOR=HEAD─────────┤
   └─,SETCURSOR=TAIL─────────┘

►──┬─,KEYRANGE=NO_CHANGE──────────────────────────────────────────────────────┬────►
   └─,KEYRANGE=SET─,KEYRANGESTART=keyrangestart─,KEYRANGEEND=keyrangeend──────┘

►──┬─,KEYRANGESTATE=NO_CHANGE─────────────────────────────────────────┬────────────►
   └─,KEYRANGESTATE=DEFINE─,KREMPTY=krempty─,KRNOTEMPTY=krnotempty────┘

►──┬─,LISTSTATE=NO_CHANGE─────────────────────────────────────────────────┬───────►◄
   └─,LISTSTATE=DEFINE─,LISTEMPTY=listempty─,LISTNOTEMPTY=listnotempty────┘
```

parameters-5

```
►►──┬─,ENTRYKEY=entrykey─────────┬───────────────────────────────────────────────►◄
    └─,SECONDARYKEY=secondarykey─┘
```

parameters-6

```
                                                 ┌─,DRIVEEXIT=YES─┐
►►──┬─,ACTION=START─,VECTORINDEX=vectorindex─────┼────────────────┼─┬────────────►◄
    │                                            └─,DRIVEEXIT=NO──┘ │
    └─,ACTION=STOP──────────────────────────────────────────────────┘
```

parameters-7

```
                    ┌─,NOTIFICATION=FIRST─┐
►►──┬─,ACTION=START─┼─────────────────────┼─,UNC=unc─┬───────────────────────────►◄
    │               └─,NOTIFICATION=EVERY─┘          │
    └─,ACTION=STOP───────────────────────────────────┘
```

parameters-8

```
►►─,STARTINDEX=startindex──,ENDINDEX=endindex──,MOSVECTOR=mosvector──────────────►

►─┬─,BUFLIST=buflist─┤ parameters-9 ├─,BUFNUM=bufnum─,BUFINCRNUM=bufincrnum─┬────►◄
  └─,BUFFER=buffer─┤ parameters-10 ├─,BUFSIZE=bufsize───────────────────────┘
```

parameters-9

```
                                                    ┌─,BUFALET=NO_BUFALET─┐
    ┌─,BUFADDRTYPE=VIRTUAL─┤ parameters-10 ├────────┼─────────────────────┼─┐
    │                                               └─,BUFALET=bufalet────┘ │
►►──┼───────────────────────────────────────────────────────────────────────┼────►◄
    │                           ┌─,BUFADDRSIZE=31─┐                         │
    └─,BUFADDRTYPE=REAL─────────┼─────────────────┼─────────────────────────┘
                                └─,BUFADDRSIZE=64─┘
```

parameters-10

```
                    ┌─,BUFSTGKEY=CALLERS_KEY─┐
    ┌─,PAGEABLE=YES─┼────────────────────────┼─┐
    │               └─,BUFSTGKEY=bufstgkey───┘ │
►►──┼──────────────────────────────────────────┼─────────────────────────────────►◄
    └─,PAGEABLE=NO─────────────────────────────┘
```

Parameter Descriptions

The parameter descriptions for IXLLSTC are listed in alphabetical order. Default values are underlined:

,ACTION=START
,ACTION=STOP

Depending on the IXLLSTC request type, use this input parameter to specify:

- For REQUEST=MONITOR_EVENTQ, whether event queue monitoring is to be started or stopped. Requires that the list structure be allocated in a coupling facility of CFLEVEL=3 or higher.
- For REQUEST=MONITOR_LIST, whether list monitoring is to be started or stopped.
- For REQUEST=MONITOR_KEYRANGE, whether key-range monitoring is to be started or stopped. Requires that the list structure be allocated in a coupling facility of CFLEVEL=9 or higher.

- For REQUEST=MONITOR_SUBLIST, whether sublist monitoring is to be started or stopped. Requires that the list structure be allocated in a coupling facility of CFLEVEL=3 or higher.

START
- For REQUEST=MONITOR_EVENTQ, start event queue monitoring for the connection specified by CONTOKEN for the specified event queue.
- For REQUEST=MONITOR_LIST, start monitoring for the connection specified by CONTOKEN for the list specified by LISTNUM.
- For REQUEST=MONITOR_KEYRANGE, start key-range monitoring for the connection specified by CONTOKEN for the list specified by LISTNUM.
- For REQUEST=MONITOR_SUBLIST, start sublist monitoring for the connection specified by CONTOKEN for the designated sublist.

STOP
- For REQUEST=MONITOR_EVENTQ, stop monitoring the event queue for the connection specified by CONTOKEN.
- For REQUEST=MONITOR_LIST, stop list monitoring for the connection specified by CONTOKEN for the list specified by LISTNUM.
- For REQUEST=MONITOR_KEYRANGE, stop key-range monitoring for the connection specified by CONTOKEN for the list specified by LISTNUM.
- For REQUEST=MONITOR_SUBLIST, stop monitoring of the designated sublist for the connection specified by CONTOKEN.

,ANSAREA=*ansarea*

Use this input parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

Not all fields in the answer area are applicable to all request types. Request type descriptions indicate which answer area fields are applicable for successful request completion cases. Return and reason code descriptions indicate which answer area fields are applicable for non-successful completing requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned by the request.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list designated by LISTNUM. If the specified list authority fails to equal the list authority for the designated list, the IXLLSTC operation is terminated with no resultant change to the structure. Indicative return and reason codes are provided to the invoker.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16 byte field that contains the list authority value.

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*

Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*

Use this input or output parameter to hold either input data for the request or output data from the request.

- For READ_LCONTROLS requests, the buffer must be 4096 bytes on a 4096-byte boundary.

  Upon successful completion of a READ_LCONTROLS request, the BUFFER area contains, starting at offset zero, an array of list monitoring information for the specified list and an array of key-range monitoring information for the specified list. The relative position of an array element associates it with a connection identifier. The first array element is associated with a connection identifier of zero, and is reserved. The length and contents of each array element is defined by mapping macro IXLYLMI.
- For DEQ_EVENTQ requests, the buffer must be 4096 bytes on a 4096-byte boundary.

  Upon successful completion of a DEQ_EVENTQ request, the BUFFER area contains, starting at offset zero, an array of event monitor controls that were dequeued from the event queue. The length and contents of each array element is defined by mapping macro IXLYEMC.
- For MONITOR_SUBLISTS requests, the BUFSIZE keyword specifies the size of the buffer. You can define the buffer size to be a total size of up to 65536 bytes. Depending on the size you select, the following restrictions apply:
  - If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
    - Is 256, 512, 1024, 2048, or 4096 bytes.
    - Starts on a 256-byte boundary.
    - Does not cross a 4096-byte boundary.
  - If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
    - Is a multiple of 4096 bytes.
    - Is less than or equal to 65536 bytes.
    - Starts on a 4096-byte boundary.

  For a MONITOR_SUBLISTS request, the BUFFER is used to input an array of entries, each mapped by IXLYMSRI, and each of which contains the information necessary to register as a sublist monitor for one sublist.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry data.

,BUFINCRNUM=*bufincrnum*

Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*

Use this input or output parameter to specify a list of buffers to contain input data for the request or to receive output data from the request.

The buffer address description is an 8-byte element. The first four bytes of the element is reserved space. The second four bytes of the element contains the address of the buffer.

- For READ_LCONTROLS requests, only one buffer may be passed. The buffer must be 4096 bytes in length and must start on a 4096-byte boundary.

  Upon successful completion of a READ_LCONTROLS request, the BUFLIST buffer contains, starting at offset zero, an array of list monitoring information for the specified list and an array of key-range monitoring information for the specified list. The relative position of an array element associates it with a connection identifier. The first array element is associated with a connection identifier of zero, and is reserved. The length and contents of each array element is defined by mapping macro IXLYLMI.
- For DEQ_EVENTQ requests, only one buffer may be passed. The buffer must be 4096 bytes in length on a 4096-byte boundary.

  Upon successful completion of a DEQ_EVENTQ request, the BUFLIST buffer contains, starting at offset zero, an array of event monitor controls that were dequeued from the event queue. The length and contents of each array element is defined by mapping macro IXLYEMC.
- For MONITOR_SUBLISTS requests, there may be 1 to 16 buffers in the list. Each buffer in the list must be the same size and must reside in the same address space or data space. Data is fetched from the buffers in the order specified.

  For MONITOR_SUBLISTS requests, the length of a buffer must be a multiple of 256 bytes between 256 and 4096. Each buffer must start on a 256-byte boundary and must not cross a 4096-byte boundary.

  For a MONITOR_SUBLISTS request, the BUFLIST is used to input an array of entries, each mapped by IXLYMSRI, and each of which contains the information necessary to register as a sublist monitor for one sublist.

  Note: The buffers do not have to be contiguous in storage. (List services treats BUFLIST buffers as a single, contiguous buffer, even if the buffers are not contiguous.)

See the BUFNUM and BUFINCRNUM parameter descriptions for defining the number and size of buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 1 to 16.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*
Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*
Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

CONTOKEN=*contoken*
Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DRIVEEXIT=YES
,DRIVEEXIT=NO
Depending on the IXLLSTC request type, use this input parameter to specify:

- For REQUEST=MONITOR_EVENTQ, whether XES should drive the connection's list transition exit when the state of the specified user's event queue changes from empty to not-empty.
- For REQUEST=MONITOR_LIST, whether XES should drive the connection's list transition exit when the state of a list changes from empty to not-empty.
- For REQUEST=MONITOR_KEYRANGE, whether XES should drive the connection's list transition exit when the state of the key-range changes from empty to not-empty.

YES

- For REQUEST=MONITOR_EVENTQ, when the state of the user's event queue changes from empty to not-empty, XES will drive the connection's list transition exit.
- For REQUEST=MONITOR_LIST, when the state of a list changes from empty to not-empty, XES will drive the connection's list transition exit.
- For REQUEST=MONITOR_KEYRANGE, when the state of a key-range changes from empty to not-empty, XES will drive the connection's list transition exit.

NO

- For REQUEST=MONITOR_EVENTQ, when the state of the user's event queue changes from empty to not-empty, XES will not drive the connection's list transition exit.
- For REQUEST=MONITOR_LIST, when the state of a list changes from empty to not-empty, XES will not drive the connection's list transition exit.
- For REQUEST=MONITOR_KEYRANGE, when the state of the key-range changes from empty to not-empty, XES will not drive the connection's list transition exit.

,ENDINDEX=*endindex*
For REQUEST=MONITOR_SUBLIST, use this input parameter to specify the 1-origin index number of the last IXLYMSRI entry which is requested to be processed. The valid range is from the STARTINDEX value to 1024, inclusive. The specified index must not imply a larger buffer size than the user has actually provided for the BUFFER or BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the 1-origin index number of the last entry to be processed. that contains the entry identifier.

,ENTRYKEY=*entrykey*

Depending on the IXLLSTC request type, use this input parameter to specify an unsigned 128-bit entry key to be used in conjunction with LISTNUM to designate:

- For REQUEST=READ_EMCONTROLS, a sublist whose EMCs are to be read.
- For REQUEST=MONITOR_SUBLIST, a sublist whose monitoring is to be started or stopped.

Specify ENTRYKEY only for structures that support keyed entries. The use of ENTRYKEY requires that the list structure be allocated in a coupling facility of CFLEVEL=3 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the unsigned 128-bit entry key.

,KEYRANGE=NO_CHANGE
,KEYRANGE=SET

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify whether the key-range is to be updated.

NO_CHANGE

The key-range currently defined for the list will remain unchanged.

SET

The key-range defined for the list is to be set to the range specified by KEYRANGESTART and KEYRANGEEND.

SET is valid for keyed list structures allocated in a coupling facility of CFLEVEL=9 or higher. SET is ignored and the request processed as if NO_CHANGE were specified when the structure is allocated in a coupling facility of CFLEVEL=8 or lower, or if the structure does not support keyed list entries.

The default key-range start and end key values are initialized to binary zeros for all lists when the structure is allocated.

,KEYRANGEEND=*keyrangeend*

Use this input parameter to specify an unsigned 128-bit key-range end key value to be associated with the list. The end key value must be greater than or equal to the start key value.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the key-range end key value.

,KEYRANGESTART=*keyrangestart*

Use this input parameter to specify an unsigned 128-bit key-range start key value to be associated with the list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the key-range start key value.

,KEYRANGESTATE=NO_CHANGE
,KEYRANGESTATE=DEFINE

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify whether the threshold counts that define the empty or not-empty state of a key-range are to be updated.

NO_CHANGE

The threshold counts will remain unchanged.

DEFINE

The threshold counts that define the empty or not-empty state of a key-range are to be set to the values indicated by KREMPTY and KRNOTEMPTY.

DEFINE is valid for keyed list structures allocated in a coupling facility of CFLEVEL=9 or higher. DEFINE is ignored and the request processed as if NO_CHANGE were specified when the structure is allocated in a coupling facility of CFLEVEL=8 or lower, or if the structure does not support keyed list entries.

A key-range is either in the empty state or the not-empty state. Use a REQUEST=MONITOR_KEYRANGE request to register interest in monitoring transitions between key-range states.

The default key-range empty or not-empty threshold counts are initialized to zero for all lists when the structure is allocated.

A request to define the key-range empty or not-empty threshold counts can timeout. The caller is expected to reissue the request until it completes without timing out.

,KEYTYPE=ENTRY
,KEYTYPE=SECONDARY

Depending on the IXLLSTC request type, use this input parameter to specify:

- For REQUEST=READ_EQCONTROLS, whether to return information about the event queue for state transitions of sublists identified by list entry key or to return information about the event queue for state transitions of sublists identified by secondary key.
- For REQUEST=DEQ_EVENTQ, whether to dequeue EMCs for state transitions of sublists identified by list entry key or to dequeue EMCs for state transitions of sublists identified by secondary key.
- For REQUEST=MONITOR_EVENTQ, whether to monitor the event queue for state transitions of sublists identified by list entry key or to monitor the event queue for state transitions of sublists identified by secondary key.

ENTRY

- For REQUEST=READ_EQCONTROLS, return information about the event queue for state transitions of sublists identified by list entry key.
- For REQUEST=DEQ_EVENTQ, dequeue the EMCs for state transitions of sublists identified by list entry.
- For REQUEST=MONITOR_EVENTQ, monitor the event queue for state transitions of sublists identified by list entry key.

KEYTYPE=ENTRY is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher.

SECONDARY

- For REQUEST=READ_EQCONTROLS, return information about the event queue for state transitions of sublists identified by secondary key.
- For REQUEST=DEQ_EVENTQ, dequeue the EMCs for state transitions of sublists identified by secondary key.
- For REQUEST=MONITOR_EVENTQ, monitor the event queue for state transitions of sublists identified by secondary key.

KEYTYPE=SECONDARY is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

,KREMPTY=*krempty*

Use this input parameter to specify the key-range empty threshold count to be associated with the list.

A key-range is in the empty state if the number of list entries in the key-range is either less than or equal to the KREMPTY threshold or zero. Upon completion of the key-range state definition, the key-range is also considered to be in the empty state if the number of list entries in the key-range is less than or equal to the KRNOTEMPTY threshold. Once the key-range state becomes empty, it remains empty until the number of list entries in the key-range becomes greater than the KRNOTEMPTY threshold.

The KREMPTY keyword is valid only for a keyed list structure allocated in a coupling facility of CFLEVEL=9 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the key-range empty threshold count to be associated with the list.

,KRNOTEMPTY=*krnotempty*

Use this input parameter to specify the key-range not-empty threshold count to be associated with the list. The key-range not-empty count must be greater than or equal to the key-range empty count.

A key-range is in the not-empty state if the number of list entries in the key-range is greater than the KRNOTEMPTY threshold. Once the key-range state is not-empty, it remains not-empty until the number of list entries in the key-range either becomes less than or equal to the KREMPTY threshold or becomes zero.

The KRNOTEMPTY keyword is valid only for a keyed list structure allocated in a coupling facility of CFLEVEL=9 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the key-range not-empty threshold to be associated with the list.

,LISTDESC=NO_LISTDESC
,LISTDESC=*listdesc*

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify the user-defined description to be associated with the list. If LISTDESC is not specified, the user description for the designated list will remain unchanged.

Note: The default list description for all lists when the structure is allocated is binary zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte field that contains the user-defined description.

,LISTEMPTY=*listempty*

Use this input parameter to specify the list empty threshold count to be associated with the list.

A list is in the empty state if the number of list entries on the list is either zero or less than or equal to the LISTEMPTY threshold.

The LISTEMPTY keyword is valid only for a list structure allocated in a coupling facility of CFLEVEL=9 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list empty threshold count.

,LISTKEY=NO_LISTKEY
,LISTKEY=*listkey*

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify the list key to be associated with the list. The list key value may be assigned to list entries when they are created or moved.

If LISTKEY is not specified, the list key for the designated list will remain unchanged. LISTKEY is ignored when the structure does not support keyed entries.

Note: The default list key for all lists when the structure is allocated is binary zeros.

The LISTKEY keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the list key to be associated with the list.

,LISTLIMIT=NO_LISTLIMIT
,LISTLIMIT=*listlimit*

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify the list limit bounding the number of entries or elements that can reside on the list. If LISTLIMIT is not specified, the list limit for the designated list will remain unchanged.

Note: The default list limit for all lists when the structure is allocated is the total number of list entries or elements in the structure.

When an IXLALTER is processed for a list structure, if the list limit for a list is equal to the default value then it will be automatically adjusted to the new number of entries or elements. If a list limit is not equal to the default value then the alter process will not adjust it and it is the responsibility of the user, if desired, to set a new limit taking into account the current entry and element counts for the altered structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list limit for the list.

,LISTNOTEMPTY=*listnotempty*

Use this input parameter to specify the list not-empty threshold count to be associated with the list. The list not-empty threshold count must be greater than or equal to the list empty threshold count.

A list is in the not-empty state if the number of list entries on the list is greater than the LISTNOTEMPTY threshold.

The LISTNOTEMPTY keyword is valid only for a list structure allocated in a coupling facility of CFLEVEL=9 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list not-empty threshold count.

,LISTNUM=NO_LISTNUM
,LISTNUM=*listnum*

Depending on the IXLLSTC request type, use this input parameter to specify the number of the list to be processed.

- For REQUEST=READ_LCONTROLS, the number of the list to be processed.
- For REQUEST=WRITE_LCONTROLS, the number of the list to be processed.
- For REQUEST=READ_EMCONTROLS, the partial designation of the sublist to be processed.

  LISTNUM is used in conjunction with either ENTRYKEY or SECONDARYKEY to designate a sublist for which the user’s event monitor controls are to be read.
- For REQUEST=MONITOR_LIST, the number of the list to be monitored.
- For REQUEST=MONITOR_KEYRANGE, the number of the list whose key-range is to be monitored.
- For REQUEST=MONITOR_SUBLIST, the partial designation of the sublist to be processed.

  LISTNUM may be used in conjunction with ENTRYKEY or SECONDARYKEY to designate a sublist for which the user wishes to start or stop sublist monitoring.

To Code: Specify the RS-type name or address (using a 2 register from 2 to 12) of a fullword field that contains the appropriate list or sublist value.

,LISTSTATE=NO_CHANGE
,LISTSTATE=DEFINE

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify whether the threshold counts that define the empty or not-empty state of a list are to be updated.

NO_CHANGE

The threshold counts will remain unchanged.

DEFINE

The threshold counts are to be set to the values indicated by LISTEMPTY and LISTNOTEMPTY.

Upon completion of the definition of the list empty or not-empty thresholds, the list state may change. If the number of list entries is less than the new list empty thresholds, the list state is empty. If the number of list entries is greater than the new list not-empty threshold, the list state is not-empty. When the number of list entries is greater than or equal to the new list empty threshold and less than or equal to the new list not-empty threshold, the list state remains unchanged.

DEFINE is valid for coupling facilities of CFLEVEL=9 or higher. DEFINE is ignored and the request processed as if NO_CHANGE were specified when the structure is allocated in a coupling facility of CFLEVEL=8 or lower.

A list is either in the empty state or the not-empty state. Use a REQUEST=MONITOR_LIST request to register interest in monitoring transitions between list states.

The default list empty or not-empty threshold counts are initialized to zero for all lists when the structure is allocated. For structures allocated in a coupling facility of CFLEVEL=8 or lower, the list-empty and list-not-empty threshold counts are always zero.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of this request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description of the LOCKOPER values to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA
,LOCKDATA=*lockdata*

Use this input parameter to specify user information that is to be passed to the Notify exit for the connection when the lock is held because of a LOCKOPER=SET operation and another connection issues one of the following requests:

- An IXLLSTC LOCK request specifying LOCKOPER=SET with LOCKMODE=UNCOND
- An IXLLSTE request specifying LOCKOPER=SET with LOCKMODE=UNCOND
- An IXLLSTM request specifying LOCKOPER=NOTHELD

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-specified information.

,LOCKINDEX=*lockindex*

For REQUEST=LOCK, use this input parameter to specify the index of the lock within the lock table for the list structure that is to be operated on as specified by the LOCKOPER keyword. The index value must fall within the range 0 to the number of lock table entries minus one, inclusive.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the lock index.

,LOCKMODE=UNCOND
,LOCKMODE=COND

Use this input parameter to specify how contention on the lock is to be handled.

UNCOND
The lock operation will be performed unconditionally. If the specified lock is held, the IXLLSTC request will be held up until the operation can be processed.

- If MODE=SYNCSUSPEND is specified, the caller is suspended until the lock becomes available.
- If any MODE value other than SYNCSUSPEND is specified, the request is processed asynchronously and the caller is notified by the means indicated on the MODE keyword.

COND
The lock operation is performed conditionally. If the lock is held, the IXLLSTC request is terminated with no resultant change to the structure, and indicative return and reason codes are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=TEST
,LOCKOPER=READNEXT
For REQUEST=LOCK, use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET

- When LOCKCOMP is not specified, requests ownership of the lock for the connection specified by CONTOKEN.
- When LOCKCOMP is specified, requests that ownership of the lock be transferred from the connection specified by LOCKCOMP to the connection specified by CONTOKEN. Any outstanding requests awaiting contention resolution on this lock are ignored.

RESET

- When LOCKCOMP is not specified, requests that ownership of the lock be released if held by the connection specified by CONTOKEN.
- When LOCKCOMP is specified, requests that ownership of the lock be released if it is held by the connection specified by LOCKCOMP.

The RESET operation will always be done unconditionally when LOCKCOMP is not specified and conditionally when LOCKCOMP is specified.

TEST

- When LOCKCOMP is not specified, requests that the lock be tested to see if it is owned by the connection specified by CONTOKEN.
- When LOCKCOMP is specified, requests that the lock be tested to see if it is owned by the LOCKCOMP connection.

In both cases return code X'0' indicates that the lock was held and return code X'4' indicates that the lock is either not held or is held by a different connection.

The lock state remains unchanged as a result of this option.

READNEXT

- When LOCKCOMP is not specified, requests that the lock index and connection ID of the owner of the next owned lock starting at the lock index specified by LOCKINDEX be returned.
- When LOCKCOMP is specified, requests that the lock index of the next lock owned by the LOCKCOMP connection, starting at the lock index specified by LOCKINDEX, be returned.

The lock state remains unchanged as a result of this option.

A request specifying LOCKOPER=READNEXT may complete prematurely if coupling facility model-dependent timeout criteria is exceeded. In this event, indicative return and reason codes are provided, and the index of the next lock to be processed is returned in the answer area specified by ANSAREA. This lock index can be specified on a subsequent LOCKOPER=READNEXT request to resume processing with the appropriate lock entry.

,MAXLISTKEY=NO_MAXLISTKEY
,MAXLISTKEY=*maxlistkey*

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify the maximum list key value to be associated with the list. This value specifies an upper bound for the list key value.

If MAXLISTKEY is not specified, the maximum list key for the designated list will remain unchanged. MAXLISTKEY is ignored when the structure does not support keyed entries.

Note: The default maximum list key for all lists when the structure is allocated is binary zeros.

The MAXLISTKEY keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the maximum list key value to be associated with the list.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT

,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify whether the request is to be performed synchronously or asynchronously.

SYNCSUSPEND

The request will be performed synchronously. Control is not returned to the caller until request processing is complete and the final disposition determined.

If necessary, the caller will be suspended until the request completes. The caller must be executing in an enabled state to use this option.

SYNCECB

The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request, and the ECB specified by REQECB is posted when the request has completed.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the REQECB ECB is posted.

SYNCEXIT

The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request. When the request completes, the connection's Complete Exit will be called.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the connection's Complete Exit is called.

SYNCTOKEN

The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request and a token that uniquely identifies the request is returned. This token must be specified on a subsequent invocation of IXLFCOMP to force completion of the request and determine its final disposition.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

SYNCECB

The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the ECB specified by REQECB will be posted.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the REQECB ECB is posted.

ASYNCEXIT

The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the connection's Complete Exit will be called.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the connection's Complete Exit is called.

ASYNCTOKEN

The request is to be initiated, a token generated that uniquely identifies the request on this system, and control returned to the caller prior to completion of the requested operation. The token must be specified on a subsequent invocation of IXLFCOMP to force completion of the request and determine its final disposition.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

ASYNCNORESPONSE

The request is to be initiated and control returned to the caller prior to completion of the requested operation. No asynchronous request token is returned, hence no external mechanism exists to force completion of the request.

MODE=ASYNCNORESPONSE is mutually exclusive with LOCK, READ_LCONTROLS, DEQ_EVENTQ, and MONITOR_SUBLISTS requests. It is also mutually exclusive with REQUEST=MONITOR_LIST and REQUEST=MONITOR_KEYRANGE when ACTION=START is specified. Any other request can specify MODE=ASYNCNORESPONSE.

,MOSVECTOR=*mosvector*

Use this input parameter to specify a 1-origin bit string in which each bit represents the monitored object state (empty or not-empty) of a particular sublist at the time that the REQUEST=MONITOR_SUBLISTS request was processed.

The MOSVECTOR bits correspond one-to-one with the IXLYMSRI entries that were passed as input in the BUFFER or BUFLIST. Only the bits corresponding to the IXLYMSRI entries that were actually processed on the current request (that is, between STARTINDEX and ENDINDEX for a request that completed successfully, or between STARTINDEX and the returned index of the first unprocessed entry minus one for premature completion cases) will contain valid monitored object state information for the sublists designated by the corresponding IXLYMSRI entries. Bits in the MOSVECTOR that lie outside the valid range are not meaningful.

When a bit in the valid range is on, the sublist designated by the corresponding IXLYMSRI entry was not-empty at the time the request was processed. When a bit in the valid range is off, the sublist designated by the corresponding IXLYMSRI entry was empty at the time the request was processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte field that contains a 1-origin bit string in which each bit represents the monitored object state of a particular sublist.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*

For REQUEST=WRITE_LCONTROLS, use this input parameter to specify a new value to be established as the list authority of the designated list. If NEWAUTH is not specified, the list authority for the designated list will remain unchanged.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the list authority value.

,NOTIFICATION=FIRST
,NOTIFICATION=EVERY

For REQUEST=MONITOR_SUBLIST, use this input parameter to specify the condition for which the coupling facility is to queue an EMC to the registered user's event queue.

FIRST

Queue the EMC to the registered user's event queue when the monitored sublist transitions from empty to not-empty.

EVERY

Queue the EMC to the registered user's event queue whenever a list entry is queued to the monitored sublist.

NOTIFICATION=EVERY is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

,PAGEABLE=YES

,PAGEABLE=NO

Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST reside in pageable storage.

YES

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage.

This includes disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) It does not include implicitly non-pageable storage (such as is obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO

Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage.

This includes implicitly non-pageable storage areas. If the virtual storage may potentially become pageable, the invoker is responsible for ensuring the virtual storage remains non-pageable for the duration of the request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a list request.)

If MODE=ASYNCTOKEN is specified or MODE=SYNCTOKEN is specified and the request does not complete synchronously, the storage must remain non-pageable until completion of the corresponding IXLFCOMP request. If MODE=ASYNCEXIT is specified or MODE=SYNCEXIT is specified and the request does not complete synchronously, the storage must remain non-pageable until the complete exit is driven for the request. If MODE=ASYNCECB is specified or MODE=SYNCECB is specified and the request does not complete synchronously, the storage must remain non-pageable until the specified ECB is posted for the request.

The system takes responsibility for managing binds to central storage for the duration of the list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of a list request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLLSTC Version Support" on page 1023 for a description of the options available with the PLISTVER macro.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,REQUEST=LOCK
,REQUEST=READ_LCONTROLS
,REQUEST=WRITE_LCONTROLS
,REQUEST=READ_EQCONTROLS
,REQUEST=READ_EMCONTROLS
,REQUEST=DEQ_EVENTQ
,REQUEST=MONITOR_EVENTQ
,REQUEST=MONITOR_LIST
,REQUEST=MONITOR_KEYRANGE
,REQUEST=MONITOR_SUBLIST
,REQUEST=MONITOR_SUBLISTS

Use this input parameter to specify the type of operation to be performed on the structure.

LOCK

The lock entry designated by LOCKINDEX is to be operated on as specified by the LOCKOPER keyword. This request type may only be specified for structures that contain a lock table.

REQUEST=LOCK is mutually exclusive with MODE=ASYNCNORESPONSE.

READ_LCONTROLS

Read the control information for the list specified by the LISTNUM keyword. The information returned in the answer area specified by ANSAREA consists of the list controls as mapped by the IXLYLAA macro. In addition, the list monitoring information and key-range monitoring information for the list, as mapped by the IXLYLMI macro, is returned in the storage specified by BUFFER or the buffers specified by BUFLIST.

REQUEST=READ_LCONTROLS is mutually exclusive with MODE=ASYNCNORESPONSE.

WRITE_LCONTROLS
Update one or more of the list controls for the list specified by LISTNUM. The list controls that can be updated include:

- List authority (NEWAUTH)
- List limit bounding the number of entries or elements that may reside on the list (LISTLIMIT)
- List descriptor (LISTDESC)
- List key (LISTKEY)
- Maximum list key (MAXLISTKEY)
- List cursor and list cursor direction (SETCURSOR)
- Key-range start and end values (KEYRANGESTART and KEYRANGEEND)
- Key-range empty and key-range not-empty threshold counts (KREMPTY and KRNOTEMPTY)
- List empty and list not-empty threshold counts (LISTEMPTY and LISTNOTEMPTY

READ_EQCONTROLS
Read event queue control information for the requesting user's event queue. Event queue processing is used in conjunction with sublist monitoring and requires that the list structure be allocated in a coupling facility of CFLEVEL=3 or higher. For list structures allocated in a coupling facility of CFLEVEL=9 or higher, a sublist can be identified either by entry key or by secondary key. Event monitor controls (EMCs) for sublists identified by entry key are queued to a different event queue than EMCs for sublists identified by secondary key. The different event queues can be monitored independently. The KEYTYPE keyword designates which event queue is the subject of the READ_EQCONTROLS request.

Information returned in the answer area specified by ANSAREA consists of the following:

- The total number of event monitor controls (EMCs) currently on the event queue
- The number of times that the event queue has transitioned from the empty to the non-empty state
- The vector index number that is being used to monitor the event queue
- An indication of whether or not the user's list transition exit is to be driven when the event queue transitions from the empty to the not-empty state
- An indication of whether the EMCs are for sublists identified by entry key or for sublists identified by secondary key.

READ_EMCONTROLS
Read the EMC control information for the requesting user's registered interest in monitoring a particular sublist designated by the LISTNUM keyword and either ENTRYKEY or SECONDARYKEY.

- For entry keys, this request type is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher. The structure must support keyed list entries (REFOPTION=KEY must be specified on the IXLCONN request when the structure is allocated).
- For secondary keys, this request type is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher. The structure must support secondary keys (KEYTYPE=SECONDARY must be specified on the IXLCONN request when the structure is allocated).

Information returned in the answer area specified by ANSAREA includes the following:

- The user notification controls (UNC)
- The list number and entry key or secondary key of the sublist with which the event monitor control information is associated
- An indication of whether or not the EMC is currently queued to the user's event queue

- An indication of whether the EMCs are for sublists identified by entry key or for sublists identified by secondary key
- An indication of whether the EMC is queued to the event queue for only the first list entry added to the sublist or for every list entry added to the sublist.

DEQ_EVENTQ

Dequeue the event monitor controls from an event queue. The sublist associated with the event queue can be identified either by list entry key or by secondary key. Identification by list entry key requires that the list structure be allocated in a coupling facility of CFLEVEL=3 or higher; identification by secondary key requires that the list structure be allocated in a coupling facility of CFLEVEL=9 or higher. If the structure is allocated with secondary keys, there are two event queues that can be monitored. The KEYTYPE keyword designates which event queue is the subject of the DEQ_EVENTQ request.

The contents of the dequeued EMCs are returned in the specified BUFFER or BUFLIST area. Each individual EMC is mapped by the IXLYEMC macro. Note that neither the EMCs nor the sublist monitoring interest which they represent is deleted by this request. Sublist monitoring remains active until it is stopped by a subsequent MONITOR_SUBLIST ACTION=STOP request.

This request type is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher (for entry keys) or CFLEVEL=9 or higher (for secondary keys).

The DEQ_EVENTQ request may complete prematurely, that is, without having dequeued all of the EMCs from an event queue. In such a case, the user is expected to process the EMCs that were returned on the current request and then re-issue the DEQ_EVENTQ request to continue the process of dequeueing the remaining EMCs from the event queue.

Information returned in the answer area specified by ANSAREA consists of the number of EMCs that were returned and the number of EMCs that still remain on the user’s event queue.

REQUEST=DEQ_EVENTQ is mutually exclusive with MODE=ASYNCNORESPONSE.

MONITOR_EVENTQ

Start or stop list notification vector monitoring of an event queue for a requesting user.

This request type is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher. The user must have a local vector associated with the connection to the structure, and the structure must support keyed list entries. If the structure is allocated in a coupling facility of CFLEVEL=9 or higher and is allocated with secondary keys, there are two event queues that can be monitored. The KEYTYPE keyword designates which event queue is the subject of the MONITOR_EVENTQ request.

While event queue monitoring is in effect for an event queue, the IXLVECTR service can be used to determine whether the event queue contains any event monitor controls (EMCs). These EMCs represent not-empty sublists in which the user has registered sublist monitoring interest. (See “IXLVECTR — Check or Modify Local Cache Vector or List Notification Vector” on page 1221.) Use event queue monitoring in conjunction with sublist monitoring (MONITOR_SUBLIST or MONITOR_SUBLISTS) to process state transitions for monitored sublists. Use the DEQ_EVENTQ request to dequeue queued EMCs from an event queue and to read the contents of the EMCs.

If a list notification vector index is in use for monitoring, a request to stop monitoring should be performed before using the same vector index to start another monitor. If event queue monitoring is in effect, a request to start monitoring the same event queue with a different VECTORINDEX or DRIVEEXIT specification will cause the old specifications to be immediately replaced by the new specifications. It is not necessary to stop event queue monitoring before requesting to start monitoring the same event queue using the new specifications. However, because the replaced vector index is no longer being used to monitor an event queue, it may be reassigned for other uses (for example, to monitor a list).

When event queue monitoring is stopped, the state of the vector index that was previously being used to monitor the event queue is not defined. However, because the vector index is no longer being used to monitor the event queue, it may be reassigned for other uses (for example, to monitor a list).

MONITOR_LIST
Start or stop monitoring the list specified by LISTNUM.

While list monitoring is in effect for a list, the IXLVECTR service can be used to determine whether the list is considered empty or not-empty according to the thresholds established with a WRITE_LCONTROLS request.

The IXLVECTR service can also be used to alter the size of the list notification vector, and thus changed the number of lists, key-ranges, or event queues that can be monitored concurrently. (See “IXLVECTR — Check or Modify Local Cache Vector or List Notification Vector” on page 1221.)

If a list notification vector index is in use for monitoring, a request to stop monitoring should be performed before using the same vector index to start another monitor. If a vector index is in use for monitoring a list, a request to start list monitoring for the same list with a different vector index will cause the vector index in use for the list to be replaced by the new vector index. In this case, it is not necessary to stop monitoring using the old index before requesting to start monitoring using the new index.

REQUEST=MONITOR_LIST with ACTION=START is mutually exclusive with MODE=ASYNCNORESPONSE.

REQUEST=WRITE_LCONTROLS can be used to set the empty or not-empty threshold counts that are used to define the empty or not-empty state of a list for list monitoring.

MONITOR_KEYRANGE
Start or stop key-range monitoring of the list specified by LISTNUM. This request type is valid only for keyed list structures allocated in a coupling facility of CFLEVEL=9 or higher.

While key-range monitoring is in effect for a list, the IXLVECTR service can be used to determine whether the key-range is considered empty or not-empty according to the thresholds established with a WRITE_LCONTROLS request that specifies KEYRANGESTATE=DEFINE.

The IXLVECTR service can also be used to alter the size of the list notification vector, and thus change the number of lists, key-ranges, or event queues that can be monitored concurrently. (See “IXLVECTR — Check or Modify Local Cache Vector or List Notification Vector” on page 1221.)

If a list notification vector index is in use for monitoring, a request to stop monitoring should be performed before using the same vector index to start another monitor. If a vector index is in use for monitoring a key-range, a request to start key-range monitoring of the same list with a different vector index will cause the vector index in use for the key-range to be replaced by the new vector index. In this case, it is not necessary to stop monitoring using the old index before equesting to start monitoring using the new index.

Only one key-range can be monitored per list. REQUEST=WRITE_LCONTROLS can be used to set the starting and ending key-range values, and the empty or not-empty threshold counts that define the empty or not-empty state of a key-range for key-range monitoring.

REQUEST=MONITOR_KEYRANGE with ACTION=START is mutually exclusive with MODE=ASYNCNORESPONSE.

A request to start monitoring a key-range for a list can timeout if definition of the key-range is not complete (see REQUEST=WRITE_LCONTROLS with KEYRANGESTATE=DEFINE). The caller is expected to reissue the request until it completes without timing out.

MONITOR_SUBLIST
Start or stop monitoring a sublist designated by a list number and either an entry key or secondary key within the list structure.

A request to monitor a keyed sublist is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher. The user must have a local vector associated with the connection to the structure, and the structure must support keyed list entries (REFOPTION=KEY must be specified on IXLCONN when the structure is allocated).

A request to monitor a secondary keyed sublist is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher. The user must have a local vector associated with the connection to the structure, and the structure must support secondary keys (KEYTYPE=SECONDARY must be specified on IXLCONN when the structure is allocated).

MONITOR_SUBLISTS
Start monitoring a set of sublists designated by list number and either entry key or secondary key in a list structure. MONITOR_SUBLISTS cannot be used to stop sublist monitoring.

A request to monitor a sublist identified by a list entry key is valid only when the structure is allocated in a coupling facility of CFLEVEL=3 or higher. A request to monitor a sublist identified by a secondary key is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher and the structure supports secondary keys. In both cases, the user must have a local vector associated with the connection to the structure, and the structure must support keyed list entries.

REQUEST=MONITOR_SUBLISTS is mutually exclusive with MODE=ASYNCNORESPONSE.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,SECONDARYKEY=*secondarykey*
Depending on the IXLLSTC request type, use this input parameter to specify the unsigned 256-bit secondary key to be used in conjunction with LISTNUM to designate:

- For REQUEST=READ_EMCONTROLS, a sublist whose EMCs are to be read.
- For REQUEST=MONITOR_SUBLIST, a sublist whose monitoring is to be started or stopped.

SECONDARYKEY is valid only for structures that are allocated in a coupling facility of CFLEVEL=9 or higher. The structure must support secondary keys.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-byte field that contains the unsigned 256-bit secondary key.

,SETCURSOR=NO_SETCURSOR
,SETCURSOR=HEAD
,SETCURSOR=TAIL
For REQUEST=WRITE_LCONTROLS, use this input parameter to specify that the list cursor and the list cursor direction are to be set.

The list cursor direction is only used when an IXLLSTE invocation specifies UPDATECURSOR=YES with CURSORUPDATETYPE=NEXTCOND.

Note: The default list cursor for all lists when the structure is allocated is binary zeros. The default list cursor direction for all lists when the structure is allocated is a head-to-tail direction.

The SETCURSOR keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

NO_SETCURSOR
The list cursor and the list cursor direction will remain unchanged.

HEAD
The list cursor is to be set to the entry identifier for the first entry on the list and the list cursor direction is to be set in a head-to-tail direction. If the list is empty, the list cursor will be reset to binary zeros.

TAIL
The list cursor is to be set to the entry identifier for the last entry on the list and the list cursor direction is to be set in a tail-to-head direction. If the list is empty, the list cursor will be reset to binary zeros.

,STARTINDEX=*startindex*
For REQUEST=MONITOR_SUBLISTS, use this input parameter to specify the 1-origin index number of the first IXLYMSRI entry that is requested to be processed. The valid range is from 1 to the ENDINDEX value, inclusive.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the 1-origin index number of the first entry to be processed.

,UNC=*unc*
For REQUEST=MONITOR_SUBLIST, use this input parameter to specify the user notication controls (UNC) that represent the user’s monitoring interest in the designated sublist. The user notification control information resides in the event monitor controls (EMC), which is queued to an event queue when the monitored sublist becomes not-empty, and is returned along with other information from the EMC on a REQUEST=DEQ_EVENTQ.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the user notification controls (UNC).

,VECTORINDEX=*vectorindex*
Depending on the IXLLSTC request, use this input parameter to specify a vector index.

- For REQUEST=MONITOR_EVENTQ, specify the vector index to be associated with the monitored event queue. If the request completes successfully, this local vector index in the vector for the connection specified by CONTOKEN will subsequently reflect the empty or not-empty state of the user’s event queue.
- For REQUEST=MONITOR_LIST, specify the vector index that is to reflect the empty or not-empty state of the monitored list. If the request completes successfully, this local notification vector index in the vector for the connection specified by CONTOKEN will subsequently reflect the empty or not-empty state of the monitored list.
- For REQUEST=MONITOR_KEYRANGE, specify the vector index that is to reflect the empty or not-empty state of the monitored key-range. If the request completes successfully, this local notification vector index in the vector for the connection specified by CONTOKEN will subsequently reflect the empty or not-empty state of the monitored key-range.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list notification vector index.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.

- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 70. Return and Reason Codes for IXLLSTC Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The IXLLSTC request completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • For a MONITOR_SUBLISTS request, the index of the first unprocessed IXLYMSRI entry. All prior IXLYMSRI entries were processed. • For a DEQ_EVENTQ request, the number of EMCs that were dequeued from the user's event queue, and the number of EMCs that remain on the user's event queue after dequeueing those EMCs. • For a LOCK request, the index of the next lock to be processed. For WRITE_LCONTROLS request and MONITOR_KEYRANGE requests, no specific information is returned in the answer area. **Action:** After processing all returned data, reissue the request to continue. For more information about premature completion, of an IXLLSTC request, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040E	**Equate Symbol:** IXLRSNCODELOCKNOTHELD **Meaning:** A LOCKOPER=TEST request determined that the specified lock was not held for the specified connection. The connection ID of the lock owner is returned in the answer area (field LAACONID). **Action:** None necessary. If this reason code is not expected, determine who holds the lock and see if any clean-up is necessary. The lock may need to be released, or you can wait and try again.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For a request that specified either LOCKOPER=READNEXT or LOCKOPER=TEST, the request could not be completed successfully because the specified lock is not generally available. **Action:** Retry the request, or obtain the lock as required, and retry the request.
4	xxxx041F	**Equate Symbol:** IXLRSNCODENOLOCKSHELD **Meaning:** A request specifying LOCKOPER=READNEXT found no locks held from the LOCKINDEX lock to the end of the lock table. **Action:** None necessary.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated event monitor controls object (EMC) does not exist for the user of the designated sublist. **Action:** None necessary. However, if this return code and reason code are unexpected, you should examine the parameters specified for the list number and entry key on the invocation of this macro.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The value specified for either STARTINDEX or ENDINDEX was not valid. No entries were processed. The index of the first entry that was not processed is returned in the answer area. **Action:** Ensure that the STARTINDEX and ENDINDEX values are valid and resubmit the request.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLIST while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLIST macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTC while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTC macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLSTC while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLSTC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTC macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying LOCKOPER=SET with LOCKMODE=UNCOND, or LOCKOPER=NOTHELD with LOCKMODE=UNCOND failed because the lock is held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLSTC request type either required the structure to support entry keys or was a monitor request type that required a keyed structure. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0852	**Equate Symbol:** IXLRSNCODENOLISTVECTOR **Meaning:** Program error. The request failed because no local vector for monitoring list headers, key ranges, or event queues exists for this connection. Either a list notification vector was not requested for this connection, or the list notification vector has been deleted. **Action:** Either you are not connected to this structure or the VECTORLEN parameter was not specified when the IXLCONN was done for this structure.
8	xxxx0853	**Equate Symbol:** IXLRSNCODEINVLISTVINDEX **Meaning:** Program error. The list notification vector index (VECTORINDEX) specified with ACTION=START was not valid. This might be because the vector index you specified is greater than the number of vector entries in the list notification vector. **Action:**Correct the VECTORINDEX that you are using.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority value for the specified list does not meet the criteria specified by AUTHCOMP. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLSTC is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size, and reissue the request. For READ_LCONTROLS and DEQ_EVENTQ requests, the specified buffer area must be 4096 bytes in length.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect. For requests that write coupling facility data, the data cannot be fetched from the specified buffer area. For requests that read coupling facility data, the data cannot be stored into the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLSTC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTC macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx0880	**Equate Symbol:** IXLRSNCODEBADMOSVECTOR **Meaning:** Program error. The storage area specified by MOSVECTOR is not addressable. **Action:** Ensure that: • The correct storage area address was used. • If you are running in AR-mode and the MOSVECTOR was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLSTC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTC macro.
8	xxxx0891	**Equate Symbol:** IXLRSNCODEBADKEYRANGEEND **Meaning:** Program error. The specified KEYRANGEEND value is not valid. The value of KEYRANGEEND must be greater than or equal to the value of KEYRANGESTART. **Action:** Ensure that the value specified by KEYRANGEEND is valid.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0892	**Equate Symbol:** IXLRSNCODEBADKRNOTEMPTY **Meaning:** Program error. The specified KRNOTEMPTY value is not valid. The value of KRNOTEMPTY must be greater than or equal to the value of KREMPTY. **Action:** Ensure that the value specified by KRNOTEMPTY is valid.
8	xxxx0893	**Equate Symbol:** IXLRSNCODEBADLISTNOTEMPTY **Meaning:** Program error. The specified LISTNOTEMPTY value is not valid. The value of LISTNOTEMPTY must be greater than or equal to the value of LISTEMPTY. **Action:** Ensure that the value specified for LISTNOTEMPTY is greater than or equal to the value of LISTEMPTY.
8	xxxx0897	**Equate Symbol:** IXLRSNCODEBADKEYTYPE **Meaning:** Program error. The specified KEYTYPE value is not suitable for the structure. KEYTYPE=SECONDARY is valid only if the structure supports secondary keys. KEYTYPE=ENTRY is valid only if the structure supports entry keys. **Action:** Correct the value of KEYTYPE.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The MONITOR_SUBLIST or MONITOR_SUBLISTS request attempted to create a new event monitor controls (EMC) object, but the structure is full and cannot accommodate any more EMCs. For a MONITOR_SUBLISTS request, the index of the IXLYMSRI entry that experienced the structure full condition is returned in the answer area. **Action:** Determine why the structure is full. • For a MONITOR_SUBLIST request, you should be monitoring the use of EMC objects (LAAMNSL_EMCCNT is the count of EMCs is use when sublist monitoring was established and LAAMNSL_MAXEMCCNT is the maximum number of EMCs for the structure.) Evaluate this data periodically to ensure structure resources are being used efficiently. You might be able to delete existing EMCs to free up space, or, if rebuild is allowed (IXLCONN macro, ALLOWREBLD parameter), rebuild the structure either to make it larger or with a changed EMCSTGPCT to allow more EMCs to be created. • For a MONITOR_SUBLISTS request, you should be monitoring the usage of the structure every time a REQUEST=MONITOR_SUBLISTS is done (LAAMNSLS_EMCCNT is the total count of EMCs in use in the list structure and LAAMNSLS_MAXEMCCNT is the maximum number of EMCs for the list structure.). This data should be evaluated periodically to ensure structure resources are being used efficiently. Field LAAMNSLS_FAILINDEX contains the index of the entry in IXLYMSRI that experienced the structure full condition. IXLYMSRI entries prior to LAAMNSLS_FAILINDEX were processed successfully.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.

Table 70. Return and Reason Codes for IXLLSTC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLSTE — XES List Structure Single Entry Services

Description

The IXLLSTE service enables you to atomically operate on individual list entries in a list structure. Some functions of IXLLSTE require that the structure be allocated in a coupling facility of CFLEVEL=9 or higher. The following services are available for operating on individual list entries:

- Create a new list entry, either by reading it from your storage area or by moving it from an existing list in the structure.
- Read a list entry into your storage area.
- Write a list entry from your storage area to a list structure.
- Move a list entry from one list to another, which may result in the creation of a new list entry.
- Delete a list entry from a list.

The IXLLSTE macro provides list services equivalent to the following IXLLIST request types:
- IXLLIST REQUEST=READ
- IXLLIST REQUEST=WRITE
- IXLLIST REQUEST=MOVE
- IXLLIST REQUEST=DELETE

See the descriptions of the corresponding IXLLIST macro for basic information about the services provided.

List structures allocated in a coupling facility of CFLEVEL=9 or higher can be allocated with certain attributes not available in a lower level coupling facility.
- Users can assign a user-designated entry identifier to a newly created list entry, rather than having the system assign the entry identifier.
- Secondary keys can be specified, thus allowing the user to reference a keyed list entry by entry key, secondary key, or both.

The IXLLSTE macro can be used to exploit these new attributes.

See *z/OS MVS Programming: Sysplex Services Guide* for information about using the IXLLSTE macro.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The current primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued for the structure.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Disabled callers must be legally disabled by holding the CPU lock and cannot hold other disabled locks. Enabled callers must not hold any locks. When MODE=SYNCSUSPEND is specified, the caller must be enabled.
Control parameters:	See “Restrictions” on page 1060

Programming Requirements

- If your program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLLSTE. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.
- Include the IXLYCON mapping macro in your program. This macro provides a list of equate symbols for users of XES services and exits.
- Include mapping macros IXLYLAA and IXLYLCTL in your program as necessary. Table 71 lists these macros, the information and areas they map, and the particular IXLLSTE requests and parameters they apply to.

Table 71. Mapping Macros for IXLLSTE

Mapping Macro	Information	Area	Request/Parameter
IXLYLAA	Answer area output	ANSAREA area	All requests
IXLYLCTL	List entry controls	Field LAALCTL of IXLYLAA	READ, WRITE, MOVE, DELETE

Restrictions

- If you specify BUFLIST and PAGEABLE=YES, all of the buffers in the list must be in the same area of storage; you cannot mix common and private storage addresses for the buffers in the list.
- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).
- The caller’s parameter list must be addressable in the caller’s primary address space.
- If the caller is running in AR ASC mode and specifies a parameter using explicit register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage areas specified by the ADJAREA and ANSAREA parameters must be addressable in the caller’s primary address space or from the caller’s PASN access list.
- The virtual storage areas specified by parameters other than ADJAREA and ANSAREA can be addressable in the caller’s primary, secondary, or home address spaces, from the PASN access list, or from the dispatchable unit access list (DU-AL).
- If the caller is disabled then the parameter list and all storage areas addressed by the macro parameters must reside in either nonpageable or disabled reference storage.

Input Register Information

Before issuing the IXLLSTE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0	Reason code
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLLSTE Version Support

The IXLLSTE macro supports versions 0, 1, and 4 — the version number corresponds to the level of the IXLLIST or IXLLSTE macro in which a keyword or function was introduced. The higher the version number, the more recent the version of the macro. Some IXLLSTE keywords are supported for multiple versions, but have different functions for each version.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLLSTE macro.
- The following IXLLSTE keywords are supported by all versions starting with version 1 and higher of the IXLLSTE macro:

LISTKEYINC

- The following keywords and functions are supported by all versions starting with version 4 and higher of the IXLLSTE macro:

ASSIGNENTRYID	LISTLIMIT	SKEYPOSITION
KEYCOMPARE	SECONDARYKEY	SKEYREQTYPE
KEYPOSITION	SKEYCOMPARE	SKEYTARGETDIR

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Summary of Version-Dependent Parameter Functions

The following table summarizes the allowed use of some of the specifications that can be made depending on the PLISTVER value specified.

Table 72. IXLLSTE Version Support

Keyword	Version	Notes
ASSIGN	0	NONE, NAME, or KEY may be specified.
	1	NONE, NAME, KEY, or LISTKEY may be specified.
ASSIGNLISTKEY	0	NO may be specified.
	1	NO, CREATE, MOVE, or ANY may be specified.
AUTHCOMPARE	0	NO may be specified.
	1	NO or YES may be specified.
CURSORUPDTYPE	0	NEXT may be specified.
	1	NEXT, NEXTCOND, CURRENT, or CURRENTCOND may be specified.
KEYTYPE	0	ENTRY may be specified.
	1	Same as version 0.
	2	Same as version 1.
	3	Same as version 2.
	4	ENTRY or SECONDARY may be specified.
MOVETOKEY	0	UNCHANGED or TARGETENTRYKEY may be specified.
	1	UNCHANGED, TARGETENTRYKEY, or LISTKEY may be specified.

Table 72. IXLLSTE Version Support (continued)

Keyword	Version	Notes
NEWAUTH	0	NO_NEWAUTH may be specified.
	1	May be specified.
VERSCOMPTYPE	0	EQUAL may be specified.
	1	EQUAL or LESSOREQUAL may be specified.

Syntax Diagram

The syntax diagram for IXLLSTE is as follows:

main diagram

```
                                       ┌─,REQID=NO_REQID─┐
►►──IXLLSTE──b──,CONTOKEN=contoken─────┼─────────────────┼──,ENTRYTYPE=──┬─NEW─┤ parameters-1 ├─┬──────────►
                                       └─,REQID=reqid────┘               ├─OLD─┤ parameters-2 ├─┤
                                                                         └─ANY─┤ parameters-3 ├─┘

  ┌─,LOCKINDEX=NO_LOCKINDEX──────────────────────┐
►─┼──────────────────────────────────────────────┼─────────────────────────────────────────────────────────►
  └─,LOCKINDEX=lockindex─┤ parameters-4 ├────────┘

  ┌─,MODE=SYNCSUSPEND─,ANSAREA=ansarea─,ANSLEN=anslen──────────────────────────────┐
►─┼────────────────────────────────────────────────────────────────────────────────┼─┬──────────────────┬──►
  ├─,MODE=SYNCECB─,REQECB=reqecb─,ANSAREA=ansarea─,ANSLEN=anslen───────────────────┤ └─,RETCODE=retcode─┘
  │                  ┌─,REQDATA=NO_REQDATA─┐                                       │
  ├─,MODE=SYNCEXIT───┼─────────────────────┼─,ANSAREA=ansarea─,ANSLEN=anslen──────┤
  │                  └─,REQDATA=reqdata────┘                                       │
  ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─,ANSAREA=ansarea─,ANSLEN=anslen─────────────┤
  ├─,MODE=ASYNCECB─,REQECB=reqecb─,ANSAREA=ansarea─,ANSLEN=anslen──────────────────┤
  │                   ┌─,REQDATA=NO_REQDATA─┐                                      │
  ├─,MODE=ASYNCEXIT───┼─────────────────────┼─,ANSAREA=ansarea─,ANSLEN=anslen─────┤
  │                   └─,REQDATA=reqdata────┘                                      │
  ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken─,ANSAREA=ansarea─,ANSLEN=anslen────────────┤
  └─,MODE=ASYNCNORESPONSE──────────────────────────────────────────────────────────┘

                       ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S─────────────────────────────────────┐
►─┬──────────────────┬─┼───────────────────────────┼──┼───────────────────────────────────────────┼──►◄
  └─,RSNCODE=rsncode─┘ ├─,PLISTVER=MAX─────────────┤  │                     ┌─,0D──────┐          │
                       └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl──────┼──────────┼─)────────┤
                                                      │                     └─,mfattr──┘          │
                                                      │                     ┌─,COMPLETE─┐         │
                                                      └─,MF=(E─,mfctrl──────┼───────────┼─)───────┘
                                                                            └─,COMPLETE─┘
```

parameters-1

```
                                         ┌─,DIRECTION=HEADTOTAIL─┐
►►──,LISTNUM=listnum──┤ parameters-6 ├───┼───────────────────────┼──────────────────────────►
                                         └─,DIRECTION=TAILTOHEAD─┘

 ┌─,ASSIGN=NONE──────────────────────────────────────────────┐  ┌─,ASSIGNENTRYID=NO_ENTRYID──┐
►┼───────────────────────────────────────────────────────────┼──┼────────────────────────────┼──►
 ├─,ASSIGN=NAME─,ENTRYNAME=entryname─────────────────────────┤  └─,ASSIGNENTRYID=assignentryid┘
 ├─,ASSIGN=KEY─,ENTRYKEY=entrykey──┤ parameters-7 ├──────────┤
 │                ┌─,LISTKEYINC=NO────────┐                  │
 └─,ASSIGN=LISTKEY┼───────────────────────┼─┤ parameters-7 ├─┘
                  └─,LISTKEYINC=listkeyinc┘

                                    ┌─,ADJAREA=NO_ADJAREA─┐
►─┤ parameters-8 ├─┤ parameters-9 ├─┼─────────────────────┼──────────────────────────────────►
                                    └─,ADJAREA=adjarea────┘

 ┌─,ELEMNUM=NO_ELEMNUM──────────────────────┐
►┼──────────────────────────────────────────┼───────────────────────────────────────────────►◄
 └─,ELEMNUM=elemnum──┤ parameters-10 ├──────┘
```

parameters-2

```
►►──,LOCATOR=──┬─CURSOR──┤ parameters-13 ├────┬─────────────────────────────────────────────►
               ├─ENTRYID──┤ parameters-14 ├───┤
               ├─ENTRYNAME──┤ parameters-15 ├─┤
               ├─UNKEYPOS──┤ parameters-16 ├──┤
               └─KEYPOS──┤ parameters-17 ├────┘

 ┌─,VERSCOMPARE=NO───────────────────────────────────────────────────┐
►┼───────────────────────────────────────────────────────────────────┼──────────────────────►
 │                                        ┌─,VERSCOMPTYPE=EQUAL──────┐│
 └─,VERSCOMPARE=YES─,VERSCOMP=verscomp────┼──────────────────────────┼┘
                                          └─,VERSCOMPTYPE=LESSOREQUAL┘

►─,REQUEST=──┬─READ──┤ parameters-18 ├───┬──────────────────────────────────────────────────►◄
             ├─WRITE──┤ parameters-19 ├──┤
             ├─MOVE──┤ parameters-20 ├───┤
             └─DELETE────────────────────┘
```

parameters-3

```
►►──,REQUEST=──┬─WRITE──┤ parameters-21 ├──┬────────────────────────────────────────────────►
               └─MOVE──┤ parameters-22 ├───┘

 ┌─,VERSCOMPARE=NO───────────────────────────────────────────────────┐
►┼───────────────────────────────────────────────────────────────────┼──┤ parameters-19 ├───►◄
 │                                        ┌─,VERSCOMPTYPE=EQUAL──────┐│
 └─,VERSCOMPARE=YES─,VERSCOMP=verscomp────┼──────────────────────────┼┘
                                          └─,VERSCOMPTYPE=LESSOREQUAL┘
```

parameters-4

```
►►─,LOCKOPER=─┬─SET─┤ parameters-5 ├──────────────────┬──────────────►◄
              │        ┌─,LOCKCOMP=NO_LOCKCOMP─┐      │
              ├─RESET──┼───────────────────────┼──────┤
              │        └─,LOCKCOMP=lockcomp────┘      │
              │          ┌─,LOCKMODE=UNCOND─┐         │
              ├─NOTHELD──┼──────────────────┼─────────┤
              │          └─,LOCKMODE=COND───┘         │
              │         ┌─,LOCKCOMP=NO_LOCKCOMP─┐     │
              └─HELDBY──┼───────────────────────┼─────┘
                        └─,LOCKCOMP=lockcomp────┘
```

parameters-5

```
   ┌─,LOCKMODE=UNCOND───────────────────────────┐ ┌─,LOCKDATA=NO_LOCKDATA─┐
►►─┼────────────────────────────────────────────┼─┼───────────────────────┼──►◄
   │                 ┌─,LOCKCOMP=NO_LOCKCOMP─┐  │ └─,LOCKDATA=lockdata────┘
   └─,LOCKMODE=COND──┼───────────────────────┼──┘
                     └─,LOCKCOMP=lockcomp────┘
```

parameters-6

```
   ┌─,AUTHCOMPARE=NO────────────────────────────────────────────────────┐ ┌─,NEWAUTH=NO_NEWAUTH─┐
►►─┼────────────────────────────────────────────────────────────────────┼─┼─────────────────────┼──►◄
   │                                        ┌─,AUTHCOMPTYPE=EQUAL───────┐ │ └─,NEWAUTH=newauth────┘
   └─,AUTHCOMPARE=YES─,AUTHCOMP=authcomp────┼───────────────────────────┼─┘
                                            └─,AUTHCOMPTYPE=LESSOREQUAL─┘
```

parameters-7

```
   ┌─,SECONDARYKEY=NO_SECONDARYKEY─┐  ┌─,SKEYTARGETDIR=HEADTOTAIL─┐
►►─┼───────────────────────────────┼──┼───────────────────────────┼──►◄
   └─,SECONDARYKEY=secondarykey────┘  └─,SKEYTARGETDIR=TAILTOHEAD─┘
```

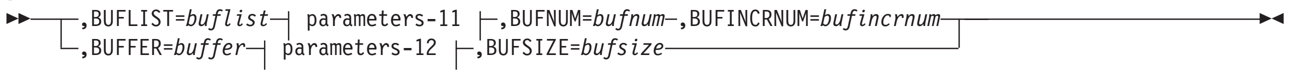
parameters-8

```
   ┌─,VERSUPDATE=NONE──────────────────┐
►►─┼───────────────────────────────────┼──►◄
   ├─,VERSUPDATE=INC───────────────────┤
   ├─,VERSUPDATE=DEC───────────────────┤
   └─,VERSUPDATE=SET─,NEWVERS=newvers──┘
```

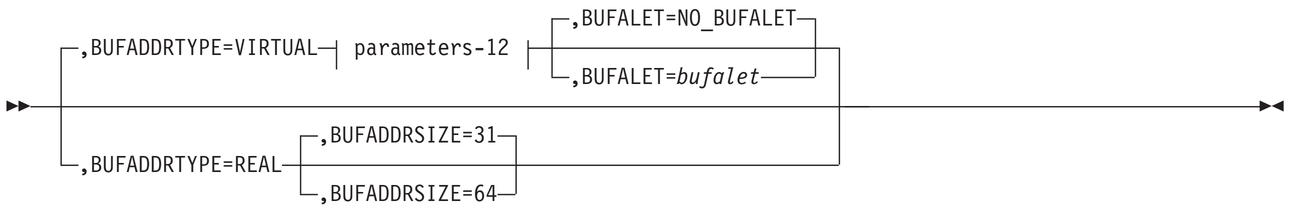
parameters-9

```
   ┌─,UPDATECURSOR=NO──────────────────────────────────────┐
►►─┼───────────────────────────────────────────────────────┼──►◄
   │                      ┌─,CURSORUPDTYPE=NEXT──────────┐ │
   └─,UPDATECURSOR=YES────┼──────────────────────────────┼─┘
                          ├─,CURSORUPDTYPE=CURRENT───────┤
                          └─,CURSORUPDTYPE=CURRENTCOND───┘
```

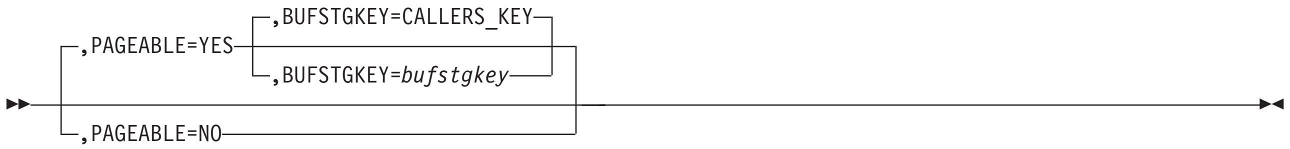
parameters-10



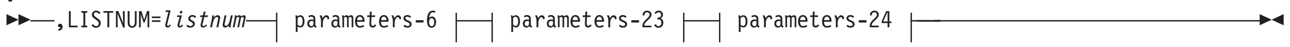
parameters-11



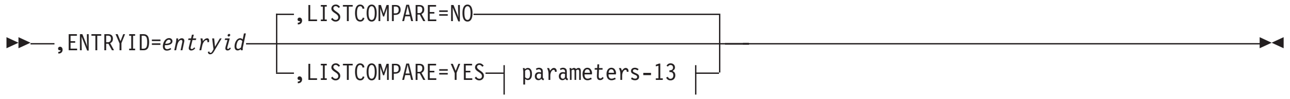
parameters-12



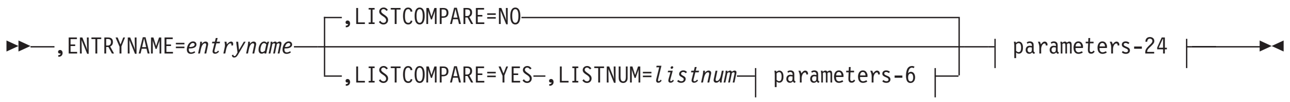
parameters-13



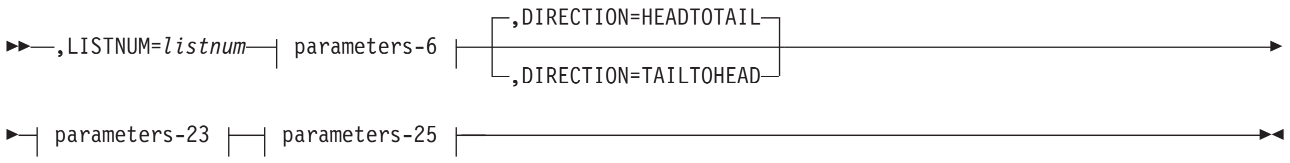
parameters-14



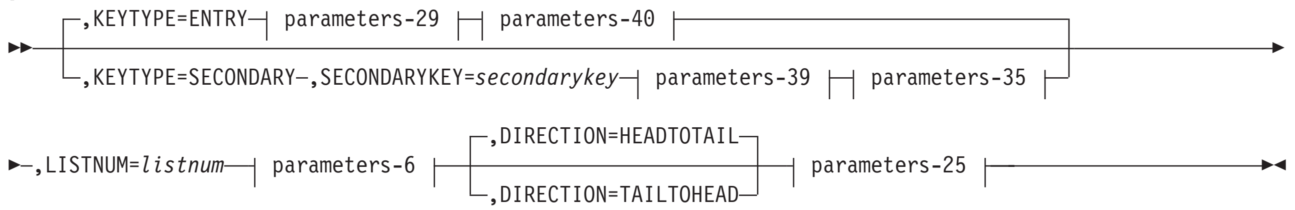
parameters-15



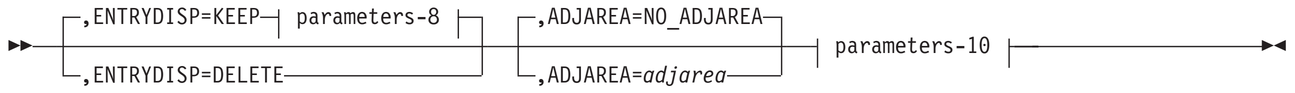
parameters-16



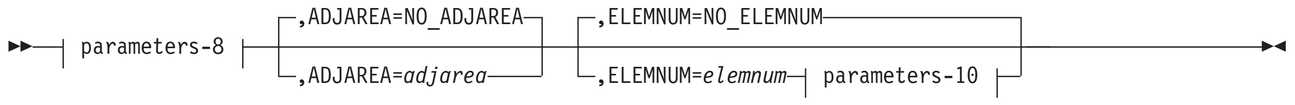
parameters-17



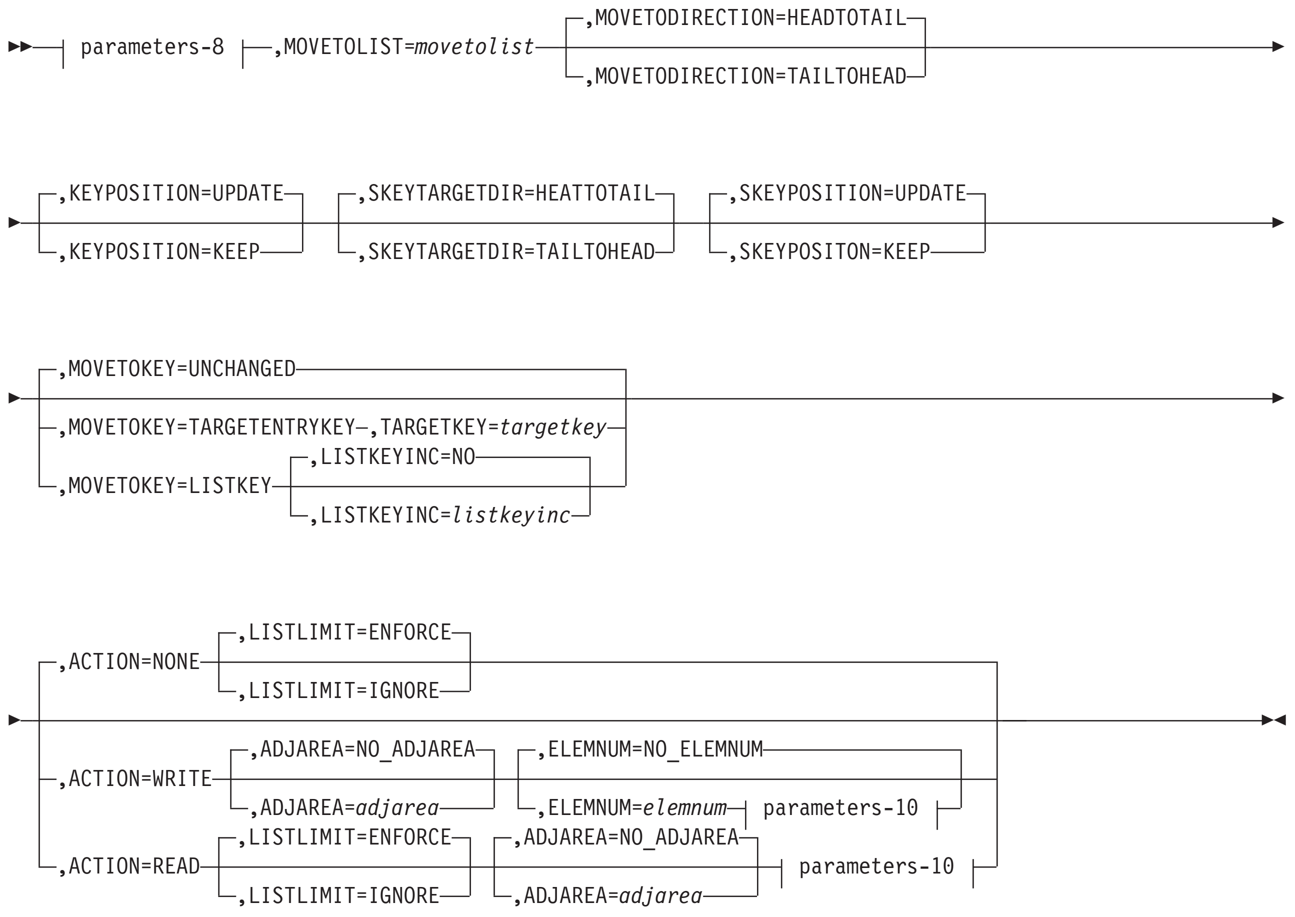
parameters-18



parameters-19



parameters-20



parameters-21

```
                                         ┌─,DIRECTION=HEADTOTAIL─┐
►►──,LISTNUM=listnum──┤ parameters-6 ├───┼───────────────────────┼──────────────────────────►
                                         └─,DIRECTION=TAILTOHEAD─┘

►─┤ parameters-9 ├─,LOCATOR=──┬─CURSOR─┤ parameters-26 ├─────┬───────────────────────────►◄
                              ├─ENTRYID─┤ parameters-27 ├────┤
                              ├─ENTRYNAME─┤ parameters-28 ├──┤
                              ├─UNKEYPOS─┤ parameters-26 ├───┤
                              └─KEYPOS─┤ parameters-30 ├─────┘
```

parameters-22

```
                            ┌─,MOVETODIRECTION=HEADTOTAIL─┐
►►──,MOVETOLIST=movetolist──┼─────────────────────────────┼─────────────────────────────►
                            └─,MOVETODIRECTION=TAILTOHEAD─┘

►─,LOCATOR=──┬─CURSOR─┤ parameters-33 ├─────┬──────────────────────────────────────────►◄
             ├─ENTRYID─┤ parameters-34 ├────┤
             ├─ENTRYNAME─┤ parameters-46 ├──┤
             ├─UNKEYPOS─┤ parameters-47 ├───┤
             └─KEYPOS─┤ parameters-37 ├─────┘
```

parameters-23

```
►►──┤ parameters-35 ├──┤ parameters-40 ├──────────────────────────────────────────────────►◄
```

parameters-24

```
    ┌─,UPDATECURSOR=NO──────────────────────────────────────────────────────┐
►►──┼───────────────────────────────────────────────────────────────────────┼─────────────►◄
    │                                                 ┌─,DIRECTION=HEADTOTAIL─┐ │
    │                     ┌─,CURSORUPDTYPE=NEXT───────┼───────────────────────┼─┐│
    │                     │                           └─,DIRECTION=TAILTOHEAD─┘ ││
    └─,UPDATECURSOR=YES───┼─────────────────────────────────────────────────────┼┘
                          ├─,CURSORUPDTYPE=NEXTCOND─────────────────────────────┤
                          ├─,CURSORUPDTYPE=CURRENT──────────────────────────────┤
                          └─,CURSORUPDTYPE=CURRENTCOND──────────────────────────┘
```

parameters-25

```
    ┌─,UPDATECURSOR=NO─────────────────────────────┐
►►──┼──────────────────────────────────────────────┼────────────────────────────────────►◄
    │                     ┌─,CURSORUPDTYPE=NEXT──────┐│
    └─,UPDATECURSOR=YES───┼──────────────────────────┼┘
                          ├─,CURSORUPDTYPE=NEXTCOND──┤
                          ├─,CURSORUPDTYPE=CURRENT───┤
                          └─,CURSORUPDTYPE=CURRENTCOND┘
```

parameters-26

```
   ┌─,ASSIGN=NONE───────────────────────────────────────────────────────────┐
►►─┼────────────────────────────────────────────────────────────────────────┼─────────────────────►
   ├─,ASSIGN=NAME─,ENTRYNAME=entryname──────────────────────────────────────┤
   ├─,ASSIGN=KEY─,ENTRYKEY=entrykey─┤ parameters-31 ├───────────────────────┤
   │                  ┌─,LISTKEYINC=NO────────────┐                         │
   └─,ASSIGN=LISTKEY──┼───────────────────────────┼─┤ parameters-32 ├───────┘
                      └─,LISTKEYINC=listkeyinc────┘

  ┌─,ASSIGNENTRYID=NO_ENTRYID─────┐
►─┼───────────────────────────────┼──────────────────────────────────────────────────────────────►◄
  └─,ASSIGNENTRYID=assignentryid──┘
```

parameters-27

```
                          ┌─,LISTCOMPARE=NO──┐
►►──,ENTRYID=entryid──────┼──────────────────┼────────────────────────────────────────────────────►
                          └─,LISTCOMPARE=YES─┘

  ┌─,ASSIGN=NONE───────────────────────────────────────────────────────────┐
►─┼────────────────────────────────────────────────────────────────────────┼─────────────────────►◄
  ├─,ASSIGN=NAME─,ENTRYNAME=entryname──────────────────────────────────────┤
  ├─,ASSIGN=KEY─,ENTRYKEY=entrykey─┤ parameters-31 ├───────────────────────┤
  │                  ┌─,LISTKEYINC=NO────────────┐                         │
  └─,ASSIGN=LISTKEY──┼───────────────────────────┼─┤ parameters-32 ├───────┘
                     └─,LISTKEYINC=listkeyinc────┘
```

parameters-28

```
                              ┌─,LISTCOMPARE=NO──┐   ┌─,ASSIGNENTRYID=NO_ENTRYID─────┐
►►──,ENTRYNAME=entryname──────┼──────────────────┼───┼───────────────────────────────┼──────────►◄
                              └─,LISTCOMPARE=YES─┘   └─,ASSIGNENTRYID=assignentryid──┘
```

parameters-29

```
►►──,ENTRYKEY=entrykey──┤ parameters-38 ├──────────────────────────────────────────────────────────►◄
```

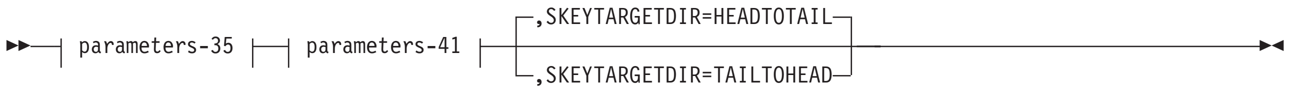
parameters-30

```
   ┌─,KEYTYPE=ENTRY─,ENTRYKEY=entrykey─┤ parameters-38 ├─┤ parameters-41 ├─┤ parameters-42 ├─┐
►►─┤                                                                                          ├──►
   └─,KEYTYPE=SECONDARY─,SECONDARYKEY=secondarykey─┤ parameters-39 ├─┤ parameters-43 ├───────┘

  ┌─,SKEYTARGETDIR=HEADTOTAIL──┐  ┌─,ASSIGNENTRYID=NO_ENTRYID─────┐
►─┼────────────────────────────┼──┼───────────────────────────────┼──────────────────────────────►◄
  └─,SKEYTARGETDIR=TAILTOHEAD──┘  └─,ASSIGNENTRYID=assignentryid──┘
```

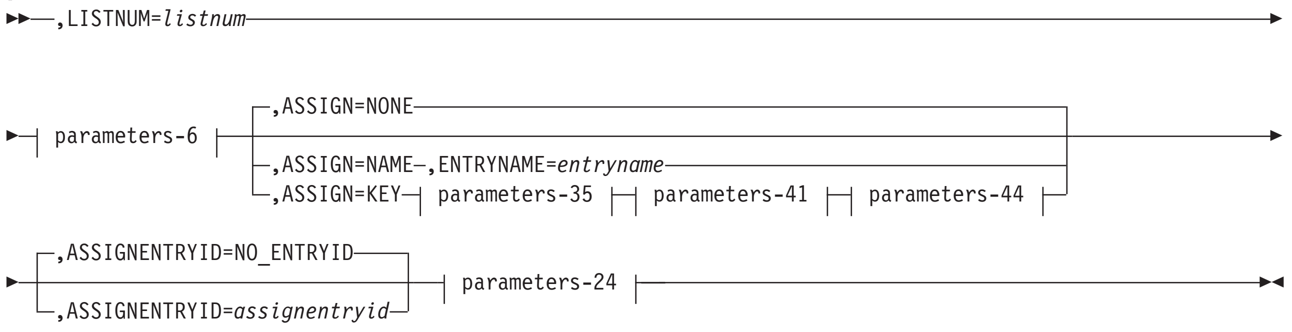
parameters-31

```
   ┌─,KEYCOMPARE=NO──────────────────────────────┐                       ┌─,SKEYTARGETDIR=HEADTOTAIL──┐
►►─┤                                             ├─┤ parameters-41 ├─────┼────────────────────────────┼───►◄
   └─,KEYCOMPARE=YES─┤ parameters-38 ├───────────┘                       └─,SKEYTARGETDIR=TAILTOHEAD──┘
```

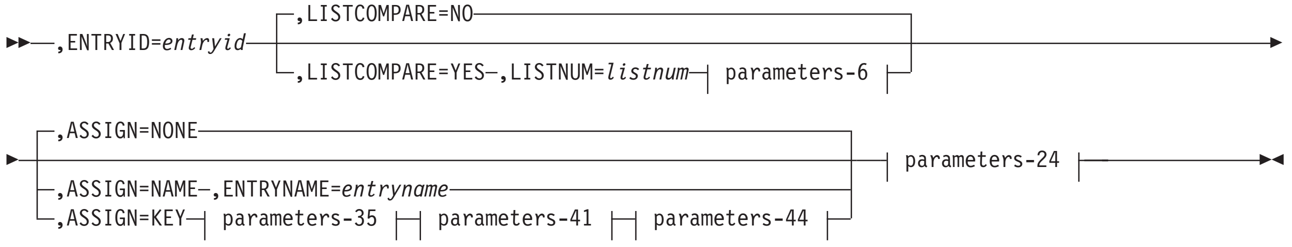
parameters-32



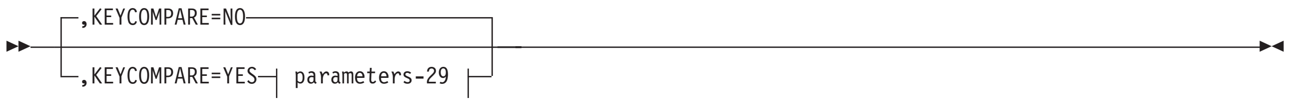
parameters-33



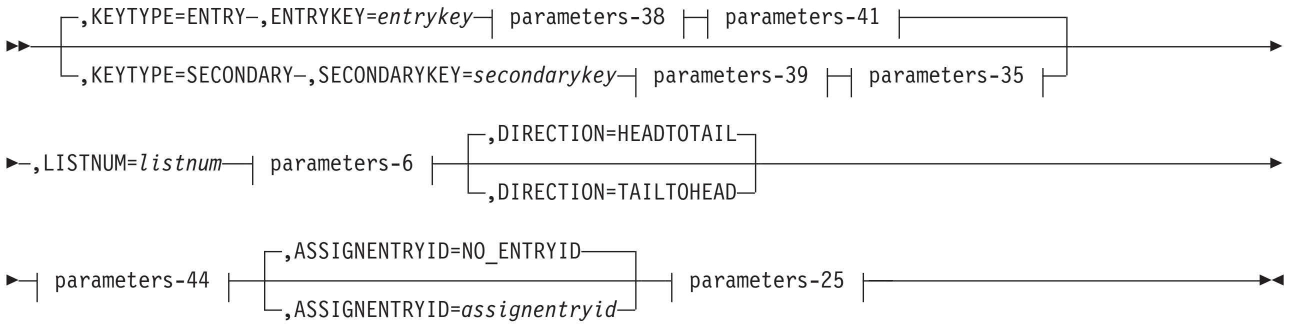
parameters-34



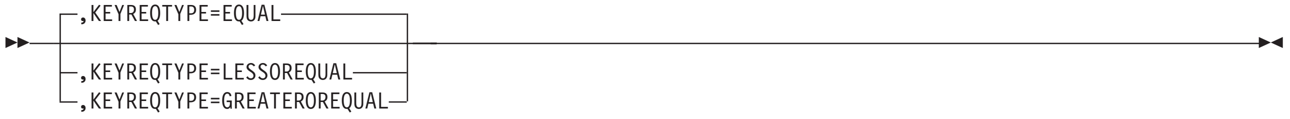
parameters-35



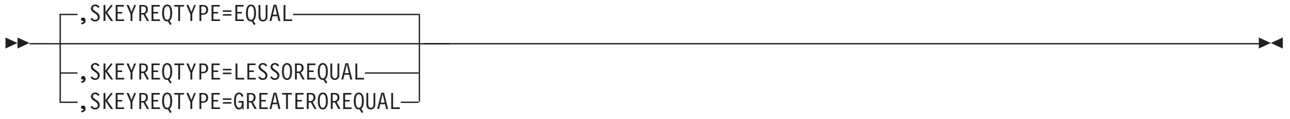
parameters-37



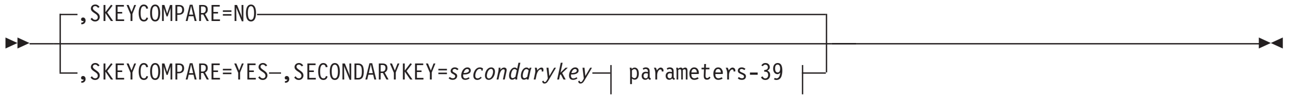
parameters-38



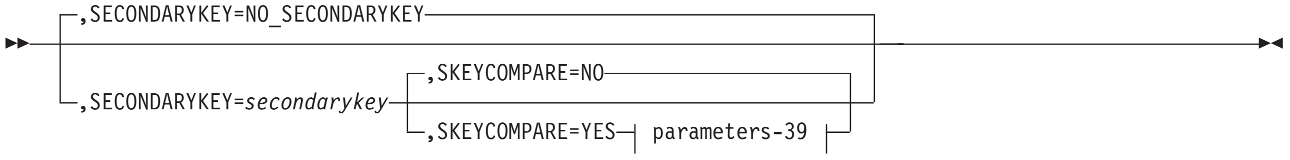
parameters-39



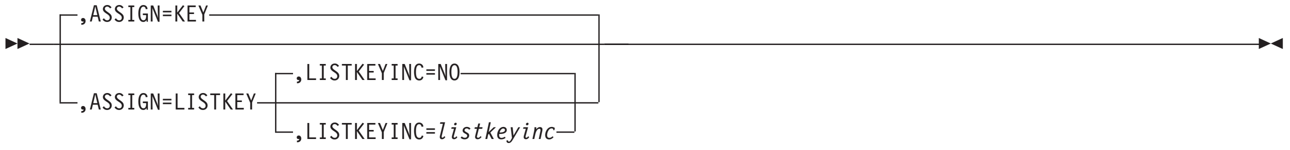
parameters-40



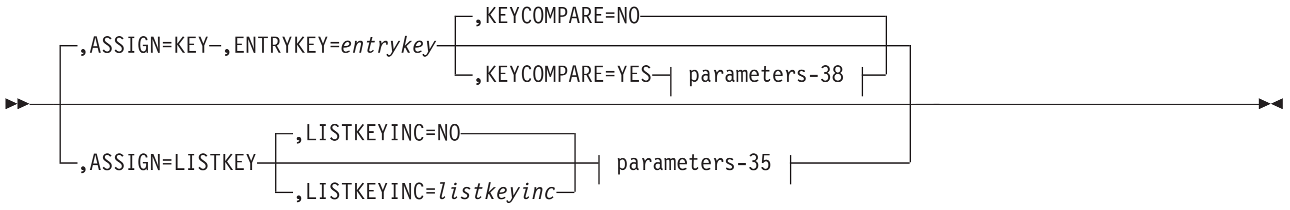
parameters-41



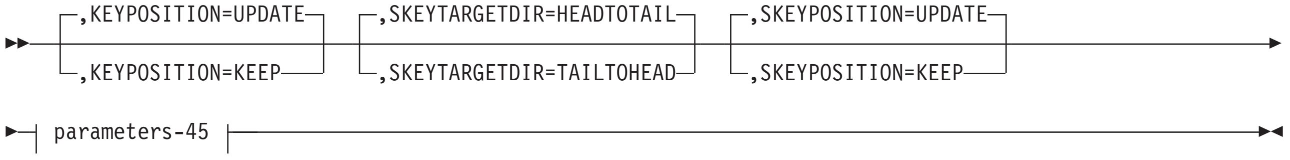
parameters-42



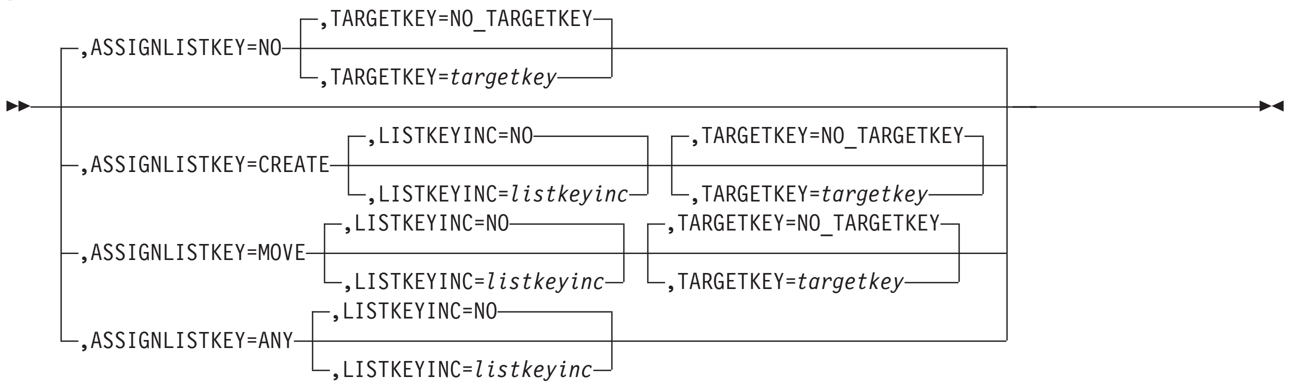
parameters-43



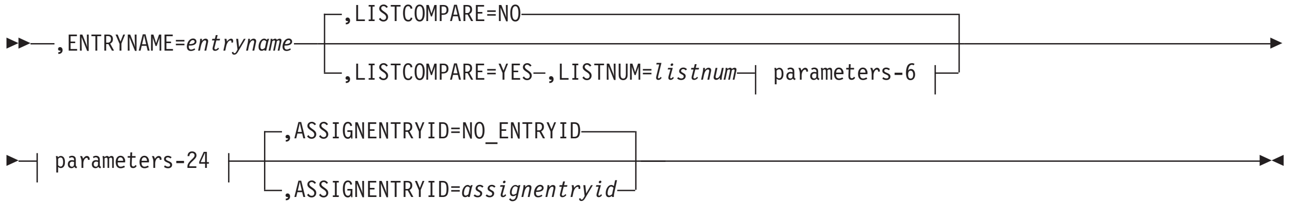
parameters-44



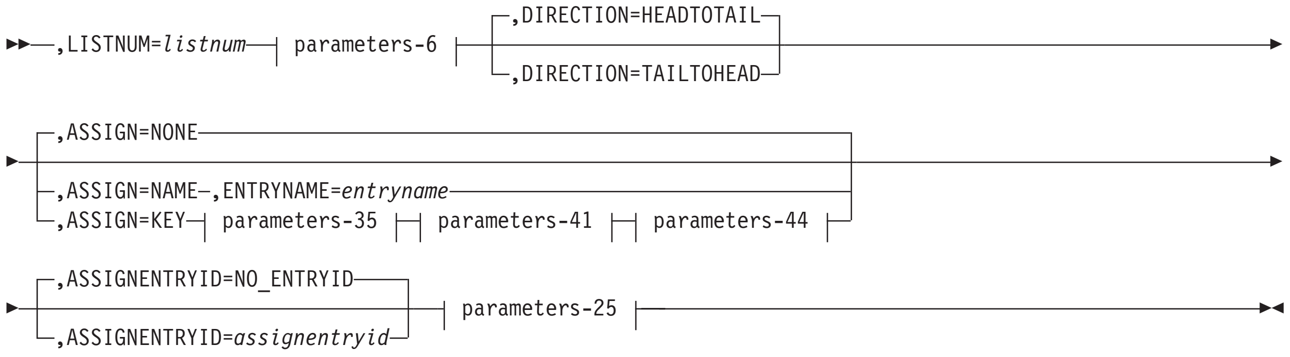
parameters-45



parameters-46



parameters-47



Parameter Descriptions

The parameter descriptions for IXLLSTE are listed in alphabetical order. Default values are underlined:

,ACTION=NONE
,ACTION=WRITE
,ACTION=READ

Use this input parameter to specify an additional action with the move request when ENTRYTYPE=OLD.

NONE
No additional action is to be performed.

WRITE
In addition to moving the list entry, write the contents of the storage area(s) specified by BUFFER or BUFLIST and/or BUFLIST to the list entry, if the storage areas exist.

READ
In addition to moving the list entry, read the entry data and/or adjunct data into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA.

At least one of BUFFER, BUFLIST, or ADJAREA must be specified when ACTION=READ is specified with REQUEST=MOVE.

,ADJAREA=NO_ADJAREA
,ADJAREA=*adjarea*
Use this input or output parameter to specify a storage area to contain the adjunct data that is read from or written to an entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

If the structure was allocated to use secondary keys, and the IXLLSTE request results in a new list entry being created, the first 32 bytes of the list entry adjunct data will be used to store the secondary key of the list entry. This allows only the last 32 bytes of list entry adjunct data to be written.

If the structure was allocated to use secondary keys and the IXLLSTE request results in the list entry being moved or updated, the first 32 bytes of the ADJAREA are ignored, allowing only the last 32 bytes of data to be written to the list entry adjunct area. The secondary key of the list entry can only be updated by an IXLLSTM REQUEST=MOVE_ENTRYLIST request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area that contains or will contain the adjunct data.

,ANSAREA=*ansarea*
Use this input parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

Not all fields in the answer area are applicable to all request types. Request type descriptions indicate which answer area fields are applicable for successful request completion cases. Return and reason code descriptions indicate which answer area fields are applicable for non-successful completing requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned by the request.

,ANSLEN=*anslen*
Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,ASSIGN=NONE
,ASSIGN=NAME
,ASSIGN=KEY
,ASSIGN=LISTKEY
Use this input parameter to specify whether to assign an ENTRYNAME, ENTRYKEY, or list key value to a newly created list entry when ENTRYTYPE=NEW or ANY. If the structure was allocated to use secondary keys, a secondary key value may be assigned to the newly created list entry.

If the list entry is moved or created, use this parameter to assign a TARGETKEY, ENTRYKEY, or list key value.

ASSIGN also allows key comparison to be specified for keyed list entries.

NONE
No assignment is to be explicitly made.

If the list entry is created, the following applies:

- If the structure was allocated to use keyed entries, the entry key (and therefore, the target list keyed position) is assigned as follows:
  - If DIRECTION=HEADTOTAIL is specified (explicitly or by default), the list entry is assigned an entry key value of all binary zeros.
  - If DIRECTION=TAILTOHEAD is specified, the list entry is assigned an entry key value of all binary ones.
- If the structure was allocated to use secondary keys, the list entry is assigned a secondary key value of all binary zeros.

If the list entry is moved, the entry key and the secondary key remain unchanged.

NAME
The entry name specified by ENTRYNAME is to be assigned to the list entry if it is created as a result of the IXLLSTE request.

If a list entry already exists with the specified ENTRYNAME, a new list entry is not created and the IXLLSTE request is ended.

ASSIGN=ENTRYNAME may only be specified when the structure was allocated to use named entries.

KEY
The entry key (and therefore the target list keyed position) specified for ENTRYKEY is to be assigned to the newly created or moved list entry.

If the list entry is created and the structure was allocated to use secondary keys, the secondary key value specified by SECONDARYKEY will be assigned to the list entry. The secondary key will be stored in the first 32 bytes of the list entry adjunct area. If the user supplies ADJAREA, the first 32 bytes will be ignored. If the user does not supply SECONDARYKEY, the secondary key will be set to all binary zeros.

For keyed list entries, when ENTRYTYPE=NEW is specified, a new list entry is created regardless of whether a list entry already exists with the same ENTRYKEY value or the same SECONDARYKEY value.

LISTKEY
The current list key value is to be assigned to the newly created list entry.

The list key and maximum list key values may be set with an IXLLSTC REQUEST=WRITE_LCONTROLS request.

The current list key value can be assigned to a list entry only on structures allocated in a coupling facility of CFLEVEL=1 or higher.

,ASSIGNENTRYID=NO_ENTRYID
,ASSIGNENTRYID=*assignentryid*
Use this input parameter to specify the ENTRYID to be assigned to the list entry created as a result of this request when ENTRYTYPE=NEW or ANY. A user provided ENTRYID must be specified if and only if ENTRYIDTYPE=USER was specified on the IXLCONN request.

If the request is attempting to create a list entry and either a list entry already exists with the specified ENTRYID, or ASSIGNENTRYID was not specified, a new list entry is not created and the IXLLSTE request is ended.

If the request is not creating a new list entry, the value specified for ASSIGNENTRYID will be ignored.

ASSIGNENTRYID is valid only when the structure is allocated in a coupling facility of CFLEVEL=8 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the ENTRYID to be assigned.

,ASSIGNLISTKEY=NO
,ASSIGNLISTKEY=CREATE
,ASSIGNLISTKEY=MOVE
,ASSIGNLISTKEY=ANY

Use this input parameter to specify how an entry key is to be assigned to the list entry if it is moved or created as a result of this request when ENTRYTYPE=ANY. If the list entry is created as a result of this request and the structure was allocated to use secondary keys, the secondary key value specified by SECONDARYKEY will be assigned to the list entry. If the user does not supply SECONDARYKEY, the secondary key will be set to all binary zeros.

The ASSIGNLISTKEY specification in conjunction with the TARGETKEY and ENTRYKEY specifications are used to assign the entry key. Note especially that the ENTRYKEY specified for KEYCOMPARE may be used as the entry key assignment for a newly created list entry depending on the ASSIGNLISTKEY specification.

The list key value and maximum list key values may be set with an IXLLSTC WRITE_LCONTROLS request.

The current list key value can be assigned to a list entry only on structures allocated in a coupling facility of CFLEVEL=1 or higher.

NO

The list key value is not to be assigned to the list entry, regardless of whether the list entry is moved or created as a result of this request. Instead, the entry key is assigned as follows:

- If the entry is created as a result of this request:
  - The list entry is assigned the TARGETKEY, if specified.
  - If TARGETKEY is not specified, the list entry is assigned the ENTRYKEY, if specified.
  - If neither TARGETKEY nor ENTRYKEY is specified, the entry key is assigned as follows:
    - If DIRECTIONHEADTOTAIL is specified (explicitly or by default), the entry key is assigned all binary zeros.
    - If DIRECTIONTAILTOHEAD is specified, the entry key is assigned all binary ones.
- If the entry is moved as a result of this request:
  - The list entry is assigned the TARGETKEY, if specified.
  - If TARGETKEY is not specified, the entry key remains unchanged.

CREATE

The list key value of the target list (and therefore, the target list keyed position) is to be assigned to the list entry if it is created as a result of this request.

If the list entry is moved as a result of this request, the entry key is assigned as follows:

- The list entry is assigned the TARGETKEY, if specified.
- If TARGETKEY is not specified, the list entry is assigned the ENTRYKEY, if specified.
- If neither TARGETKEY nor ENTRYKEY is specified, the entry key remains unchanged.

MOVE

The list key value of the target list (and therefore, the target list keyed position) is to be assigned to the list entry if it is moved as a result of this request.

If a new list entry is created as a result of this request, the entry key is assigned as follows:

- The list entry is assigned the TARGETKEY, if specified.
- If TARGETKEY is not specified, the list entry is assigned the ENTRYKEY, if specified.
- If neither TARGETKEY nor ENTRYKEY is specified, the entry key is assigned as follows:
  - If DIRECTION=HEADTOTAIL is specified (explicitly or by default), the entry key is assigned all binary zeros.
  - If DIRECTION=TAILTOHEAD is specified, the entry key is assigned all binary ones.

ANY
The list key value of the target list is to be assigned to the list entry whether the list entry is moved or created as a result of this request.

,AUTHCOMP=*authcomp*
Use this input parameter to specify an unsigned fixed 128-bit value to be compared to the list authority value for the designated list. If the comparison criterion specified by AUTHCOMPTYPE is not met, the IXLLSTE request is ended.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPARE=NO
,AUTHCOMPARE=YES
Use this input parameter to specify whether list authority comparison is to be performed to determine whether the entry should be created or processed.

The AUTHCOMPARE keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

NO
No list authority comparison is to be performed to determine whether the entry is to be created or processed.

YES
List authority comparison is to be performed to determine whether the entry is to be created or processed.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRSIZE=31
,BUFADDRSIZE=64
Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL
Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller's responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this output or input parameter to hold entry data to be read from or written to the designated entry. The BUFSIZE keyword specifies the size of the buffer.

You can define the buffer size to be a total size of up to 65536 bytes. Depending on the size you select, the following restrictions apply:

- If you specify a buffer size of less than or equal to 4096 bytes, you must ensure that the buffer:
  - Is 256, 512, 1024, 2048, or 4096 bytes.
  - Starts on a 256-byte boundary.
  - Does not cross a 4096-byte boundary.
- If you specify a buffer size of greater than 4096 bytes, you must ensure that the buffer:
  - Is a multiple of 4096 bytes.
  - Is less than or equal to 65536 bytes.
  - Starts on a 4096-byte boundary.

If BUFFER or BUFLIST is not specified, entry data is not read or written, as appropriate.

BUFFER is only functional for structures that support entry data. If the structure does not support entry data, the contents of BUFFER will not affect this request.

See the BUFSIZE parameter description for defining the size of the buffer.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry data.

,BUFINCRNUM=*bufincrnum*
Use this input parameter to specify the number of 256-byte segments comprising each buffer in the BUFLIST list.

Valid BUFINCRNUM values are 1,2,4,8, or 16, which correspond to BUFLIST buffer sizes of 256, 512, 1024, 2048, and 4096 bytes respectively.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains 1,2,4,8, or 16.

,BUFLIST=*buflist*
Use this output or input parameter to specify a 128-byte storage area containing a list of addresses of buffers that contain data to be read from or written to the designated entry.

The 128-byte storage area must:

- Consist of 0 to 16 elements.
- Each element must consist of an 8-byte field in which:
  - The left (high-order) four bytes are reserved and
  - The right (low-order) four bytes contain the address of a buffer.

The BUFLIST buffers must:

- Reside in either the same address space or data space.
- Be the same size: either 256, 512, 1024, 2048, or 4096 bytes.
- Start on a 256-byte boundary and not cross a 4096-byte boundary.

Note: The buffers do not have to be contiguous in storage. (List services treats BUFLIST buffers as a single, contiguous buffer, even if the buffers are not contiguous.)

If BUFFER or BUFLIST is not specified, entry data is not to be read or written.

BUFLIST is only functional for structures that support entry data. If the structure does not support entry data, the contents of BUFLIST will not affect this request.

See the BUFNUM and BUFINCRNUM parameter descriptions for defining the number and size of buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 0 to 16. A value of zero indicates that no data is to be read into the buffers or written to the list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller's PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The CONTOKEN uniquely identifies the user's connection to the list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,CURSORUPDTYPE=NEXT
,CURSORUPDTYPE=NEXTCOND
,CURSORUPDTYPE=CURRENT
,CURSORUPDTYPE=CURRENTCOND

Use this input parameter to specify how the list cursor is to be updated when UPDATECURSOR=YES is specified.

Note: The NEXTCOND, CURRENT, and CURRENTCOND values are valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

NEXT

Set the list cursor as follows:

- If the list entry is created, the cursor for the list specified by MOVETOLIST or LISTNUM is updated and the direction of the cursor update depends on the value specified for MOVETODIRECTION or DIRECTION.
- For an existing list entry, the cursor for the list on which the designated entry resides before the request is processed is updated and the direction of the cursor update depends on the value specified for DIRECTION.
- When MOVETODIRECTION or DIRECTION=HEADTOTAIL is specified (explicitly or by default), the cursor is set to point to the list entry following the created or processed list entry, except when it is the tail entry of the list, in which case the cursor is set to all binary zeros.
- When MOVETODIRECTION or DIRECTION=TAILTOHEAD is specified, the cursor is set to point to the list entry preceding the created or processed list entry, except when it is the head entry of the list, in which case the cursor is set to all binary zeros.

NEXTCOND

Set the list cursor as follows:

- If the list cursor points to the list entry processed for the request, and the list entry is deleted or moved to another list, set the cursor to point to the preceding or following list entry, depending on the DIRECTION specified. If the deleted or moved list entry was the last entry on the list and DIRECTION=HEADTOTAIL, or the deleted or moved list entry was the first entry on the list and DIRECTION=TAILTOHEAD, set the list cursor to binary zeros.
- If the list cursor points to the list entry processed for the request, and the list entry is moved to the same list, the cursor will remain pointing to the moved list entry.

The list cursor direction may be set by specifying SETCURSOR on an IXLLSTC WRITE_LCONTROLS request.

CURRENT

Set the list cursor as follows:

- If the list entry is created, set the list cursor to point to the created list entry.
- If the list entry is deleted or moved to another list, set the list cursor to binary zeros.
- If the list entry is moved to the same list, leave the list cursor to remain pointing to the moved list entry.
- For all other cases where the list entry is processed and remains on the same list, set the list cursor to point to the processed list entry.

CURRENTCOND

Set the list cursor as follows:

- If the list entry is created, and if the list cursor does not point to a list entry, set the list entry to point to the created list entry.
- If the list entry for the list specified for LISTNUM does not point to a list entry, and the list entry is moved to another list, the list cursor will remain binary zeros.
- If the list cursor does not point to a list entry, and the list entry is moved from and to the same list, set the cursor to point to the moved list entry.
- For all other cases where the list cursor does not point to a list entry, and the list entry is processed and remains on this same list, set the list cursor to point to the processed list entry.

If the list entry is deleted or moved to another list, then the list cursor will remain binary zeros.

,DIRECTION=<u>HEADTOTAIL</u>
,DIRECTION=TAILTOHEAD

Use this input parameter to specify the position on the list or sublist at which:

- To place the newly created list entry,

- To designate a list entry to be processed, or
- To specify the direction relative to the current entry in which the cursor is to be updated to point to the next entry on the list when CURSORUPDTYPE=NEXT is specified with UPDATECURSOR=YES.

This parameter is used in conjunction with LISTNUM, and with the assigned entry key value when keyed entries are being used.

HEADTOTAIL

Place the newly created list entry (ENTRYTYPE=NEW) according to the following:

- If the structure was not allocated to use keyed entries, the designated position is at the head of the list designated by LISTNUM.
- If the structure was allocated to use keyed entries, and ASSIGN=NONE is specified, the designated position is at the head of the list.
- If the structure was allocated to use keyed entries, and ASSIGN=KEY is specified, the designated position is at the head of the sublist designated by ENTRYKEY and LISTNUM.
- If the structure was allocated to use keyed entries, and ASSIGN=LISTKEY is specified, the designated position is at the head of the sublist designated by the list key value and LISTNUM.

Process the existing list entry (ENTRYTYPE=OLD) according to the following:

- If the structure was allocated with keyed entries and LOCATOR=KEYPOS, process the list entry at the head of the sublist designated by LISTNUM, ENTRYKEY, and KEYREQTYPE.
- If the structure was allocated with secondary keys and LOCATOR=KEYPOS, process the list entry at the head of the sublist designated by LISTNUM, SECONDARYKEY, and SKEYREQTYPE.
- If LOCATOR=UNKEYPOS is specified, process the list entry at the head of the list specified by LISTNUM.

Process the list entry (ENTRYTYPE=ANY) according to the following:

- If LOCATOR=UNKEYPOS is specified, process the list entry at the head of the list designated by LISTNUM.
- If the structure was allocated to use keyed entries and LOCATOR=KEYPOS is specified, process the list entry at the head of the sublist designated by LISTNUM, ENTRYKEY, and KEYREQTYPE.
- If the structure was allocated with secondary keys and LOCATOR=KEYPOS is specified, process the list entry at the head of the sublist designated by LISTNUM, SECONDARYKEY, and SKEYREQTYPE.
- If a list entry is created as a result of this request, and the structure was not allocated to use keyed entries, place the newly created entry at the head of the list designated by LISTNUM.
- If a list entry is created as a result of this request, and the structure was allocated to use keyed entries, place the newly created entry at the head of the sublist designated by both LISTNUM and the assigned entry key.

With UPDATECURSOR=YES, process the list cursor as follows:

- If CURSORUPDTYPE=NEXT is specified, set the cursor to point to the list entry following the processed list entry, except when the processed list entry is at the tail of the list. In that case, the cursor is set to all binary zeros.
- If CURSORUPDATE=NEXTCOND, set the cursor as follows:
  - If the list cursor points to the list entry processed for the request, and the list entry is deleted or moved to another list, set the cursor to point to the following list entry.
  - If the deleted or moved list entry was the last entry on the list, set the cursor to all binary zeros.

  - If the list cursor points to the list entry processed for the request, and the list entry is moved to the same list, the cursor will remain pointing to the moved list entry.
- If CURSORUPDATE=CURRENT, set the cursor as follows:
  - If the list entry is deleted or moved to another list, reset the list cursor to binary zeros.
  - If the list entry is moved to the same list, the cursor will remain pointing to the moved list entry.
  - For all other cases where the list entry is processed and remains on the same list, the list cursor will be set to point to the processed list entry.
- If CURSORUPDATE=CURRENTCOND, set the cursor as follows:
  - If the list cursor for the list specified by LISTNUM does not point to a list entry, and the list entry is moved to another list, the list cursor will remain binary zeros.
  - If the list cursor for the list specified for LISTNUM does not point to a list entry, and the list entry is moved from and to this same list, the cursor will be set to point to the moved list entry.
  - For all other cases where the list cursor for the list specified for LISTNUM does not point to a list entry, and the list entry is processed and remains on this same list, the list cursor will be set to point to the processed list entry.

TAILTOHEAD

Place the newly created list entry (ENTRYTYPE=NEW) according to the following:

- If the structure was not allocated to use keyed entries, the designated position is at the tail of the list designated by LISTNUM.
- If the structure was allocated to use keyed entries, and ASSIGN=NONE is specified, the designated position is at the tail of the list designated by LISTNUM.
- If the structure was allocated to use keyed entries, and ASSIGN=KEY is specified, the designated position is at the tail of the sublist designated by ENTRYKEY and LISTNUM.
- If the structure was allocated to use keyed entries, and ASSIGN=LISTKEY is specified, the designated position is at the tail of the sublist designated by the list key value and LISTNUM.

Process the existing list entry (ENTRYTYPE=OLD) according to the following:

- If the structure was allocated to use keyed entries and LOCATOR=KEYPOS, process the list entry at the tail of the sublist designated by LISTNUM, ENTRYKEY, and KEYREQTYPE.
- If the structure was allocated to use secondary keys and LOCATOR=KEYPOS, process the list entry at the tail of the sublist designated by LISTNUM, SECONDARYKEY, and SKEYREQTYPE.
- If LOCATOR=UNKEYPOS is specified, process the list entry at the tail of the list designated by LISTNUM.

Process the list entry (ENTRYTYPE=ANY) according to the following:

- If LOCATOR=UNKEYPOS is specified, process the list entry at the tail of the list designated by LISTNUM.
- If the structure was allocated to use keyed entries and LOCATOR=KEYPOS is specified, process the list entry at the tail of the sublist designated by LISTNUM, ENTRYKEY, and KEYREQTYPE.
- If the structure was allocated with secondary keys and LOCATOR=KEYPOS is specified, process the list entry at the tail of the sublist designated by LISTNUM, SECONDARYKEY, and SKEYREQTYPE.
- If a list entry is created as a result of this request, and the structure was not allocated to use keyed entries, place the newly created entry at the tail of the list designated by LISTNUM.

- If a list entry is created as a result of this request, and the structure was allocated to use keyed entries, place the newly created entry at the tail of the sublist designated by both LISTNUM and the assigned entry key.

With UPDATECURSOR=YES, process the list cursor as follows:

- If CURSORUPDTYPE=NEXT is specified, set the cursor to point to the list entry preceding the processed list entry, except when the processed list entry is at the head of the list. In that case, the cursor is set to all binary zeros.
- If CURSORUPDATE=NEXTCOND, set the cursor as follows:
  - If the list cursor points to the list entry processed for the request, and the list entry is deleted or moved to another list, set the cursor to point to the preceding list entry.
  - If the deleted or moved list entry was the first entry on the list, set the cursor to all binary zeros.
  - If the list cursor points to the list entry processed for the request, and the list entry is moved to the same list, the cursor will remain pointing to the moved list entry.
- If CURSORUPDATE=CURRENT, set the cursor as follows:
  - If the list entry is deleted or moved to another list, reset the list cursor to binary zeros.
  - If the list entry is moved to the same list, the cursor will remain pointing to the moved list entry.
  - For all other cases where the list entry is processed and remains on the same list, the list cursor will be set to point to the processed list entry.
- If CURSORUPDATE=CURRENTCOND, set the cursor as follows:
  - If the list cursor for the list specified by LISTNUM does not point to a list entry, and the list entry is moved to another list, the list cursor will remain binary zeros.
  - If the list cursor for the list specified for LISTNUM does not point to a list entry, and the list entry is moved from and to this same list, the cursor will be set to point to the moved list entry.
  - For all other cases where the list cursor for the list specified for LISTNUM does not point to a list entry, and the list entry is processed and remains on this same list, the list cursor will be set to point to the processed list entry.

,ENTRYDISP=KEEP
,ENTRYDISP=DELETE

Use this input parameter to specify whether an existing list entry when ENTRYTYPE=OLD, having been read, is to remain on the list or is to be deleted.

KEEP
The list entry is to remain on the list after the read is performed.

DELETE
The list entry is to be deleted after the read is performed.

,ELEMNUM=*elemnum*

Use this input parameter to specify the number of elements to be allocated to the existing or new data entry. Valid ELEMNUM values are from zero to the MAXELEMNUM value that was specified on the IXLCONN macro.

Considerations for specifying ELEMNUM are:

- If this request creates a new entry, the number of elements is set to the ELEMNUM value.
- If the entry already exists, the number of elements is updated to the ELEMNUM value specified.

Note: If the data from the buffers exceeds the amount of space in the elements, the data will be truncated. If the data from the buffers is less than the amount of space in the elements, the remaining space will be padded with binary zeros.

If you specify an ELEMNUM value of zero:
- No entry data will be written to the new entry.
- Any entry data in the existing entry will be deleted.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of elements.

,ENTRYID=*entryid*

Use this input parameter to specify:
- An entry identifier to be used to identify an existing list entry to be processed.
- An entry identifier to be assigned to a newly created list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=*entrykey*

Use this input parameter to designate an unsigned fixed 128-bit entry key, which can be used to:
- Specify an entry key to be used in conjunction with LISTNUM, DIRECTION, and KEYREQTYPE to locate the entry to be processed.
- Specifies an entry key to be assigned to the list entry if it is created. If there exists a sublist of one or more entries with a matching key on the list, the target position is at the head or tail of the sublist, as specified by DIRECTION. If all existing list entries have a key greater that that specified by ENTRYKEY, the target position is at the head of the list. Similarly, if the specified ENTRYKEY exceeds all of the entry keys, the target position is at the tail of the list. If none of the list entries has a matching key, and ENTRYKEY is neither the greatest nor least among the entry keys, the target position is determined according to key sequence for the list.
- For structures allocated in a coupling facility of CFLEVEL=9 or higher, when the request specifies KEYCOMPARE=YES, specifies the entry key to be compared to the entry key of the designated entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*

Use this input parameter to either:
- Assign an entry name to a new entry to be created.
- Designate an existing entry to be processed.

Specify ENTRYNAME only for structures that support named entries. Each entry name is unique within the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,ENTRYTYPE=NEW
,ENTRYTYPE=OLD
,ENTRYTYPE=ANY

Use this input parameter to specify the state in which the list entry is expected to be prior to performing the requested operation.

NEW

Write to a new list entry the contents of either the storage area specified by BUFFER or the buffers specified by BUFLIST, and the storage area specified by ADJAREA. LISTNUM designates the list number on which the created entry is to reside. The value of ASSIGN specifies whether to assign an ENTRYNAME, ENTRYKEY, or list key value to the newly created list entry. If the structure was allocated to use secondary keys, a secondary key value may be assigned to the newly created list entry.

- If the structure supports entry data and BUFFER and BUFLIST are not specified, the created entry will not contain any entry data.
- If the structure supports adjunct data and ADJAREA is not specified, the created entry will contain binary zeros for adjunct data.

When the amount of space indicated by ELEMNUM is greater than the amount of data specified for BUFFER or BUFLIST, the entry will be padded with zeros. When the amount of space indicated by ELEMNUM is less than the amount of data specified for BUFFER or BUFLIST, the input data will be truncated.

Specifying AUTHCOMPARE in conjunction with AUTHCOMP causes the list authority for the list designated by LISTNUM to precede processing. If the list authority comparison ciriterion is not met, the IXLLSTE request is terminated.

If LOCKINDEX is specified, the lock entry designated by LOCKINDEX will be operated on a specified by the LOCKOPER keyword.

The new list entry will be created as follows:

- If the structure was not allocated to use named entries or keyed entries, the newly created entry will be placed on the list specified by LISTNUM at the head or tail as specified by DIRECTION.
- If the structure was allocated to use named entries, the entry name specified for ENTRYNAME is assigned to the newly created list entry, provided a list entry does not already exist with the same entry name. The newly created entry will be placed on the list specified by LISTNUM at the head or tail as specified by DIRECTION.
- If the structure was allocated to use keyed entries, an entry key (and therefore, the target list keyed position) is assigned to the newly created list entry as follows:
  – If ASSIGN=NONE is specified, then:
    - If DIRECTION=HEADTOTAIL is specified (explicitly or by default), the newly created list entry is assigned an entry key value of all binary zeros.
    - If DIRECTION=TAILTOHEAD is specified, the newly created list entry is assigned an entry key value of all binary ones.
  – If ASSIGN=KEY is specified, the newly created list entry is assigned the value specified for ENTRYKEY.
  – If ASSIGN=LISTKEY is specified, the newly created list entry is assigned the list key value.
  – If the assigned entry key is zero, the newly created list entry is placed at the head of the list specified by LISTNUM.
  – If the assigned entry key is non-zero, the newly created list entry is placed on the list specified by LISTNUM at the head or tail of the sublist composed of list entries whose entry keys are equal to the assigned entry key. The newly created list entry is placed at the head or tail of this sublist as specified by DIRECTION. If a sublist of entries with entry keys equal to the assigned entry key does not yet exist, the newly created list entry is placed on the list in key sequence.
- If the structure was allocated to use user-provided ENTRYIDs, the newly created list entry is assigned the ENTRYID specified by ASSIGNENTRYID, provided a list entry does not already exist with the same ENTRYID.
- If the structure supports secondary keys, the secondary key specified by SECONDARYKEY will be assigned to the newly created list entry. If the user does not supply SECONDARYKEY, the newly created list entry is assigned a secondary key of all binary zeros.
  – The newly created list entry is placed on the list specified by LISTNUM at the head or tail, relative to secondary key ordering, of the sublist composed of list entries whose secondary keys are equal to the assigned secondary key. The newly created list entry is placed at the head or tail of this sublist as specified by SKEYTARGETDIR. If a sublist of entries with

secondary keys equal to the assigned secondary key does not yet exist, the target position relative to secondary key ordering is determined according to secondary key sequence for the list.

When the request completes successfully, the list entry controls, the number of entries or elements residing on the list, and the total number of allocated entries in the structure are returned in the answer area specified by ANSAREA.

OLD

Perform a read, write, move, or delete operation on an existing entry designated by the LOCATOR keyword. If the entry is not found, the requested operation is not performed and no change is made to the list structure.

Specifying AUTHCOMPARE in conjunction with AUTHCOMP causes the list authority for the list designated by LISTNUM to precede processing. If the list authority comparison ciriterion is not met, the IXLLSTE request is terminated.

If LOCKINDEX is specified, the lock entry designated by LOCKINDEX will be operated on as specified by the LOCKOPER keyword.

If the lock comparison criterion is not met, the IXLLSTE request is terminated.

In order for the request to be performed, any and all requested comparison criteria must be met. Namely, authority comparison, list number comparison, version number comparison, lock comparison, and key comparison must each succeed if requested.

ANY

Perform the specified operation on the designated entry if it exists. If the designated entry does not exist, a new entry will be created.

Specifying AUTHCOMPARE in conjunction with AUTHCOMP causes the list authority for the list designated by LISTNUM to precede processing. If the list authority comparison ciriterion is not met, the IXLLSTE request is terminated.

If LOCKINDEX is specified, the lock entry designated by LOCKINDEX will be operated on as specified by the LOCKOPER keyword.

In order for the request to be performed on an existing list entry, any and all requested comparison criteria must be met. Namely, list number comparison, version number comparison, lock comparison, and key comparison must each succeed if requested.

,KEYCOMPARE=NO
,KEYCOMPARE=YES

Use this input parameter to specify whether key comparison should be performed to determine whether the list entry should be processed.

NO

No key comparison should be performed to determine whether the list entry should be processed.

YES

Key comparison should be performed to determine whether the list entry should be processed. If the designated list entry exists but the requested key comparison criterion is not met, the IXLLSTE request is terminated.

KEYCOMPARE=YES is only meaningful when the structure is allocated in a coupling facility of CFLEVEL=9 or higher. KEYCOMPARE=YES will be ignored if the target structure was not allocated with keyed list entries.

,KEYPOSITION=UPDATE
,KEYPOSITION=KEEP

Use this input parameter to specify whether the list entry is moved from its current position on the sublist. This keyword has effect only if the list number specified by MOVETOLIST is the same as the list number on which the list entry currently resides.

The KEYPOSITION keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=9 or higher.

UPDATE
The list entry should be moved from its current position on the sublist to a position on the sublist as specified by MOVETODIRECTION and MOVETOKEY.

KEEP
The list entry should keep its current position based on entry key ordering on the sublist if and only if the list number specified by MOVETOLIST matches the current list number that contains the list entry, and the list entry is not changed by the move operation when MOVETOKEY=UNCHANGED is specified (explicitly or by default).

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL
Use this input parameter to specify how key comparison is to be performed on the designated keyed list entry or, when LOCATOR=KEYPOS is specified, to designate the list entry to be processed.

EQUAL

- The designated list entry must have a key equal to the value specified for ENTRYKEY. If the entry key is not equal to the value specified for ENTRYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that if a list entry exists with an entry key equal to the value specified for ENTRYKEY, that list entry is to be processed. If more than one such entry exists, the list entry at either the head or tail of the sublist as specified by DIRECTION is designated for processing. If no such entry exists, the IXLLSTE request is terminated.

LESSOREQUAL

- The designated list entry must have an entry key that is equal to or less than the value specified for ENTRYKEY. If the entry key is not equal to or less than the value specified for ENTRYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that:
  – If a list entry exists with an entry key equal to the value specified for ENTRYKEY, that entry is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  – If no list entries exist with an equal entry key, the list entry with the highest entry key less than the value specified for ENTRYKEY is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  – If no list entries exist with an entry key equal to or less than the value specified for ENTRYKEY, the IXLLSTE request is terminated.

GREATEROREQUAL

- The designated list entry must have an entry key that is equal to or greater than the value specified for ENTRYKEY. If the entry key is not equal to or greater than the value specified for ENTRYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that:
  – If a list entry exists with an entry key equal to the value specified for ENTRYKEY, that entry is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  – If no list entries exist with an equal entry key, the list entry with the lowest entry key greater than the value specified for ENTRYKEY is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  – If no list entries exist with an entry key equal to or greater than the value specified for ENTRYKEY, the IXLLSTE request is terminated.

,KEYTYPE=ENTRY
,KEYTYPE=SECONDARY
Use this input parameter to specify whether to locate the list entry using the list entry key or the secondary key.

ENTRY
The list entry will be located by list entry key.

SECONDARY
The list entry will be located by secondary key.

KEYTYPE=SECONDARY is valid only when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

,LISTCOMPARE=NO
,LISTCOMPARE=YES
Use this input parameter to specify whether list number comparison should be performed to determine whether the list entry should be processed.

NO
No list number comparison whould be performed to determine whether the list entry should be processed.

YES
List number comparison should be performed to determine whether the list entry should be processed.

The list number value specified for LISTNUM is compared with the list number on which the designated list entry resides. If the designated list entry exists but does not reside on the list specified by LISTNUM, the IXLLSTE request is terminated.

,LISTKEYINC=NO_LISTKEYINC
,LISTKEYINC=*listkeyinc*
Use this input parameter to specify a value to be added to the list key after the entry key is set to the list key value. If the entry key is not set to the list key value, the list key value will not be changed. If the result of adding the value specified by LISTKEYINC to the list key value is greater than the maximum list key value, the system terminates the IXLLSTE request.

Note: The LISTKEYINC parameter is valid only for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value to be added to the list key after the entry key is set to the list key value.

,LISTLIMIT=ENFORCE
,LISTLIMIT=IGNORE
Use this input parameter to specify whether the current list limits set for the target list should be enforced or ignored on a MOVE request when ENTRYTYPE=OLD. This has effect if the listlimit for the target list has been modified by an IXLLSTC WRITE_LCONTROLS request and is not equal to the default list limit.

ENFORCE
The move request will be failed if the list-entry count would exceed the current list-entry count limit of the target list or the list-element count would exceed the list-element count limit of the target list as a result of the move request.

IGNORE
The move request will proceed even if the list-entry count would exceed the current list-entry count limit of the target list or the list-element count would exceed the list-element count of the target list as a result of this move request.

,LISTNUM=NO_LISTNUM

,LISTNUM=*listnum*

Use this input parameter to specify the number of the list on which the entry to be processed currently resides. Use LISTNUM in the following ways:

- Designate the list number (when ENTRYTYPE=NEW) on which the newly created list entry should be placed.
- Check if the entry exists on the list (when ENTRYTYPE=ANY) before proceeding with the request.
- Confirm that the entry exists on the list (when ENTRYTYPE=OLD) before proceeding with the request.

To Code: Specify the RS-type name or address (using a 2 register from 2 to 12) of a 4-byte field that contains the number of the list.

,LOCATOR=CURSOR
,LOCATOR=ENTRYID
,LOCATOR=ENTRYNAME
,LOCATOR=UNKEYPOS
,LOCATOR=KEYPOS

Use this input parameter to specify the location mechanism to be used to designate the list entry to be processed.

CURSOR

The list cursor should be used to designate the list entry to be processed.

ENTRYID

The ENTRYID is to be used to designate the list entry to be processed.

ENTRYNAME

The entry name specified by ENTRYNAME is to be used to designate the list entry to be processed.

LOCATOR=ENTRYNAME may only be specified when the structure was allocated to use named entries.

UNKEYPOS

LISTNUM and DIRECTION are to be used to designate the list entry at the head or tail of the list to be processed.

KEYPOS

LISTNUM, DIRECTION, and the key specified by KEYTYPE are to be used to designate the list entry to be processed.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

This parameter has slightly different meanings based on the value specified for the LOCKOPER parameter. Generally, this input parameter specifies a connection identifier to be verified as the owner of the lock specified by LOCKINDEX. This verification is a prerequisite to the successful completion of the request.

When LOCKCOMP is specified, the completion of the request is conditional on there being no contention for the lock. If contention exists, the request will fail.

The connection identifier is available from the IXLCONN answer area, mapped by IXLYCONA, in field CONACONID.

The effect of LOCKCOMP is based on the LOCKOPER value specified. See the description under LOCKOPER to see how each request is affected by this parameter.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKDATA=NO_LOCKDATA

,LOCKDATA=*lockdata*

Use this input parameter to specify information that is to be passed to your notify exit when another user requests the LOCKINDEX lock after you have obtained the lock using LOCKOPER=SET. This user-defined information will be passed to your notify exit when the other user issues a request specifying either one of the following:

- LOCKOPER=SET
- LOCKOPER=NOTHELD with LOCKMODE=UNCOND

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined information.

,LOCKMODE=<u>UNCOND</u>
,LOCKMODE=COND

Use this input parameter to specify how contention for the lock specified by LOCKINDEX should be handled if your request causes lock contention.

UNCOND

If the specified lock is held by another user, your request is either:

- Suspended until the lock becomes available (if MODE=SYNCSUSPEND is specified).
- Performed asynchronously with notification to you when the request completes (if any MODE value other than SYNCSUSPEND is specified).

COND

The lock operation will be performed conditionally; that is, only if there is no contention for the lock. If the specified lock is held by another user, the request will be terminated with no change to the structure, and return and reason codes describing the termination are returned to the caller.

,LOCKOPER=SET
,LOCKOPER=RESET
,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the lock specified by LOCKINDEX.

SET

Set the lock.

- When LOCKCOMP is not specified, your connection gets the lock, providing no other connection holds the lock. If another connection holds the lock, the request either continues once the lock is released or fails, depending on the LOCKMODE value you specify.
- When LOCKCOMP is specified, if the connection specified by LOCKCOMP holds the lock, the lock will be transferred to your connection.

RESET

Reset the lock.

- When LOCKCOMP is not specified, the lock will be released if it is held by your connection.
- When LOCKCOMP is specified, the lock will be released if it is held by the connection specified by LOCKCOMP.

NOTHELD

The request is performed only if the lock is not held by any connection. This option also ensures that the lock remains free for the duration of the request. If another connection holds the lock, your task is suspended until the lock is released or your request fails, depending on the LOCKMODE value you specify. See the LOCKMODE description for how to handle possible lock contention.

HELDBY

Processing is determined by which connector is holding the lock.

- When LOCKCOMP is not specified, the request is performed only if your connection holds the lock.

- When LOCKCOMP is specified, the request is performed only if the lock is held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE

Use this input parameter to specify whether the request is to be performed synchronously or asynchronously.

SYNCSUSPEND

The request will be performed synchronously. Control is not returned to the caller until request processing is complete and the final disposition determined.

If necessary, the caller will be suspended until the request completes. The caller must be executing in an enabled state to use this option.

SYNCECB
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request, and the ECB specified by REQECB is posted when the request has completed.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the REQECB ECB is posted.

SYNCEXIT
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request. When the request completes, the connection's Complete Exit will be called.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the connection's Complete Exit is called.

SYNCTOKEN
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request and a token that uniquely identifies the request is returned. This token must be specified on a subsequent invocation of IXLFCOMP to force completion of the request and determine its final disposition.

If the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

SYNCECB
The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the ECB specified by REQECB will be posted.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the REQECB ECB is posted.

ASYNCEXIT
The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the connection's Complete Exit will be called.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until the connection's Complete Exit is called.

ASYNCTOKEN
The request is to be initiated, a token generated that uniquely identifies the request on this system, and control returned to the caller prior to completion of the requested operation. The token must be specified on a subsequent invocation of IXLFCOMP to force completion of the request and determine its final disposition.

Latent XES binds to the storage locations specified by BUFFER, BUFLIST, MOSVECTOR, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

ASYNCNORESPONSE
The request is to be initiated and control returned to the caller prior to completion of the requested operation. No asynchronous request token is returned, hence no external mechanism exists to force completion of the request.

MODE=ASYNCNORESPONSE is mutually exclusive with LOCKINDEX, BUFFER, and BUFLIST. Any request which does not perform a locking operation and does not use a BUFFER area or BUFLIST buffers may specify MODE=ASYNCNORESPONSE.

,MOVETODIRECTION=<u>HEADTOTAIL</u>

,MOVETODIRECTION=TAILTOHEAD

Use this input parameter in conjunction with MOVETOLIST and the entry key assigned to the list entry when keyed entries are being used, to designate the target position for the moved or created list entry.

HEADTOTAIL

One of the following:

- If the list entry is moved or created as a result of this request, and the structure was not allocated to use keyed entries, the list entry will be placed at the head of the list designated by MOVETOLIST.
- If the list entry is moved or created as a result of this request, and the structure was allocated to use keyed entries, the entry will be placed at the head of the sublist designated by MOVETOLIST and the assigned entry key.

TAILTOHEAD

One of the following:

- If the list entry is moved or created as a result of this request, and the structure was not allocated to use keyed entries, the list entry will be placed at the tail of the list designated by MOVETOLIST.
- If the list entry is moved or created as a result of this request, and the structure was allocated to use keyed entries, the entry will be placed at the tail of the sublist designated by MOVETOLIST and the assigned entry key.

,MOVETOKEY=NO_MOVETOKEY
,MOVETOKEY=*movetokey*

Use this input parameter to assign a new key, and hence the position on the MOVETOLIST, for the existing entry being moved when ENTRYTYPE=OLD. The existing entry's key will be updated to the MOVETOKEY key value. If MOVETOKEY is not specified, ENTRYKEY will remain unchanged.

Notes:

1. If there is a sublist of one or more entries with a matching key on the list then the target position is the head or tail of the sublist, as specified by the MOVETODIRECTION parameter.
2. If none of the list entries have a matching key, and MOVETOKEY is neither the greatest nor least among the list entry keys, then the target position is according to the appropriate key sequence for the list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the key.

,MOVETOLIST=*movetolist*

Use this input parameter to specify the number of the target list where the existing entry being moved or the new entry being created will be placed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the target list.

,NEWAUTH=NO_NEWAUTH
,NEWAUTH=*newauth*

Use this input parameter to specify a list authority value for the list specified by LISTNUM. If NEWAUTH is omitted, the list authority for the designated list is unchanged.

When the structure is allocated, the authority value for each list is initialized to binary zeros.

Note: The NEWAUTH parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher, except on WRITE_LCONTROLS requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,NEWVERS=*newvers*
Use this input parameter with VERSUPDATE=SET to specify an unsigned fixed 64-bit value for the list entry version number to either replace the version number of the existing entry, or initialize the version number of a new entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the entry version number.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST reside in pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage.

This includes disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) It does not include implicitly non-pageable storage (such as is obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage.

This includes implicitly non-pageable storage areas. If the virtual storage may potentially become pageable, the invoker is responsible for ensuring the virtual storage remains non-pageable for the duration of the request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a list request.)

If MODE=ASYNCTOKEN is specified or MODE=SYNCTOKEN is specified and the request does not complete synchronously, the storage must remain non-pageable until completion of the corresponding IXLFCOMP request. If MODE=ASYNCEXIT is specified or MODE=SYNCEXIT is specified and the request does not complete synchronously, the storage must remain non-pageable until the complete exit is driven for the request. If MODE=ASYNCECB is specified or MODE=SYNCECB is specified and the request does not complete synchronously, the storage must remain non-pageable until the specified ECB is posted for the request.

The system takes responsibility for managing binds to central storage for the duration of the list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of a list request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLLSTE Version Support" on page 1061 for a description of the options available with the PLISTVER macro.

,REQDATA=NO_REQDATA

,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=<u>NO_REQID</u>
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,REQUEST=READ
,REQUEST=WRITE
,REQUEST=MOVE
,REQUEST=DELETE

Use this input parameter to specify the type of operation to be performed on the designated list entry.

READ

The entry data and adjunct data and list entry controls for the designated entry are to be read.

Entry data is to be read into the storage area specified by BUFFER or the buffers specified by BUFLIST. Adjunct data is to be read into the storage area specified by ADJAREA. The absence of BUFFER, BUFLIST, and ADJAREA specifies the request is only interested in obtaining the list entry controls in the answer area specified by ANSAREA.

When the request completes successfully, the list entry controls, the number of entries or elements residing on the list, and the total number of allocated entries in the structure are returned in the answer area specified by ANSAREA.

WRITE

- For requests that specify either ENTRYTYPE=OLD or ENTRYTYPE=ANY and the designated entry already exists:

  The contents of the storage area specified by BUFFER or the buffers specified by BUFLIST, and the storage area specified by ADJAREA are to be written to the designated list entry.

  The size of list entry may be altered by specifying an ELEMNUM value that is different than the ELEMNUM value used when the entry was created. When the amount of space indicated by ELEMNUM is greater than the amount of data specified for BUFFER or BUFLIST, the entry will be padded with zeros. When the amount of spece indicated by ELEMNUM is less than the amount of data specified for BUFFER or BUFLIST, the input data will be truncated.

  When the request completes successfully, the list entry controls, the number of entries or elements residing on the list, and the total number of allocated entries in the structure are returned in the answer area specified by ANSAREA.
- For requests that specify ENTRYTYPE=ANY and the designated list entry does not exist, a new list entry will be created as follows:
  - If the structure was not allocated to use named entries or keyed entries, the newly created entry will be placed on the list specified by LISTNUM at the head or tail as specified by DIRECTION.
  - If the structure was allocated to use named entries, the entry name specified for ENTRYNAME is assigned to the newly created list entry, provided a list entry does not already exist with the same entry name. The newly created entry will be placed on the list specified by LISTNUM at the head or tail as specified by DIRECTION.
  - If the structure was allocated to use keyed entries, an entry key (and therefore, the target list keyed position) is assigned to the newly created list entry as follows:
    - If ASSIGN=NONE is specified, then:
      - If DIRECTION=HEADTOTAIL is specified (explicitly or by default), the newly created list entry is assigned an entry key value of all binary zeros.
      - If DIRECTION=TAILTOHEAD is specified, the newly created list entry is assigned an entry key value of all binary ones.
    - If ASSIGN=KEY is specified, the newly created list entry is assigned the value specified for ENTRYKEY.
    - If ASSIGN=LISTKEY is specified, the newly created list entry is assigned the list key value.
    - If the assigned entry key is zero, the newly created list entry is placed at the head of the list specified by LISTNUM.
    - If the assigned entry key is non-zero, the newly created list entry is placed on the list specified by LISTNUM at the head or tail of the sublist composed of list entries whose entry keys are equal to the assigned entry key. The newly created list entry is placed at the head or tail of this sublist as specified by DIRECTION. If a sublist of entries with entry keys equal to the assigned entry key does not yet exist, the newly created list entry is placed on the list in key sequence.
  - If the structure was allocated to use user-provided ENTRYIDs, the newly created list entry is assigned the ENTRYID specified by ASSIGNENTRYID, provided a list entry does not already exist with the same ENTRYID.
  - If the structure supports entry data and BUFFER and BUFLIST are not specified, the newly created list entry will not contain any entry data.
  - If the structure supports adjunct data and ADJAREA is not specified, the newly created list entry will contain binary zeros for adjunct data.
  - If the structure supports secondary keys, the secondary key specified by SECONDARYKEY will be assigned to the newly created list entry. If the user does not supply SECONDARYKEY, the newly created list entry is assigned a secondary key of all binary zeros.

    - The newly created list entry is placed on the list specified by LISTNUM at the head or tail, relative to secondary key ordering, of the sublist composed of list entries whose secondary keys are equal to the assigned secondary key. The newly created list entry is placed at the head or tail of this sublist as specified by SKEYTARGETDIR. If a sublist of entries with secondary keys equal to the assigned secondary key does not yet exist, the target position relative to secondary key ordering is determined according to secondary key sequence for the list.

MOVE

- When the request specifies ENTRYTYPE=OLD:
  - The designated list entry is to be moved from its current location and placed at the position specified by MOVETOLIST, MOVETODIRECTION, and if keyed entries are being used, the entry key specified by MOVETOKEY.

    When ACTION=READ and ACTION=WRITE, in addition to moving the list entry, the entry data and/or adjunct data is read into the storage areas specified by BUFFER or BUFLIST and, if applicable, ADJAREA.

    When the request completes successfully, the list entry controls, the number of entries or elements residing on the target list, and the total number of allocated entries in the structure are returned in the answer specified by ANSAREA.
  - If the structure was allocated to use keyed entries, an entry key is assigned to the moved list entry as follows:
    - If TARGETKEY is specified with MOVETOKEY=TARGETKEY, the list entry is assigned the entry key specified by TARGETKEY.
    - If MOVETOKEY=LISTKEY, the list entry is assigned the current list key value of the target list.
    - If MOVETOKEY=UNCHANGED, the entry key of the list entry remains unchanged.
- When the request specifies ENTRYTYPE=ANY and ACTION=WRITE, and the designated list entry exists:
  - In addition to moving the list entry, the data in the storage area(s) specified by BUFFER or BUFLIST and, if applicable ADJAREA, is written to the list entry. If BUFFER or BUFLIST is not specified, entry data is not to be written. If ADJAREA is not specified, structure adjunct data will not be written.
  - The size of an existing list entry may be altered by specifying an ELEMNUM value that is different from the ELEMNUM value used when the entry was created. When the amount of space indicated by ELEMNUM is greater than the amount of data specified for BUFFER or BUFLIST, the entry will be padded with zeros. When the amount of space indicated by ELEMNUM is less than the amount of data specified for BUFFER or BUFLIST, the input data will be truncated.
  - The designated list entry will be moved as follows:
    - If the structure was not allocated to use keyed entries, the list entry will be placed on the list specified by MOVETOLIST at the head or tail as specified by MOVETODIRECTION.
    - The moved list entry is placed on the list specified by MOVETOLIST at the head or tail of the sublist composed of list entries whose entry keys are equal to the assigned entry key. The moved list entry is placed at the head or tail of this sublist as specified by MOVETODIRECTION. If a sublist of entries with entry keys equal to the assigned entry key does not yet exist, the moved list entry is placed on the list in key sequence.
    - The moved list entry is placed on the list specified by MOVETOLIST at the head or tail of the sublist composed of list entries whose secondary keys are equal to the assigned secondary key. The moved list entry is placed at the head or tail of this sublist as specified by SKEYTARGETDIR. If a sublist of entries with secondary keys equal to the assigned secondary key does not yet exist, the moved list entry is placed on the list in secondary key sequence.

- If the structure was allocated to use keyed entries, an entry key (and therefore, the target list keyed position) is assigned to the list entry as follows:
  - If TARGETKEY=NO_TARGETKEY is specified (explicitly or by default) with ASSIGNLISTKEY=NO or with ASSIGNLISTKEY=CREATE, the entry key remains unchanged.
  - If TARGETKEY is specified with ASSIGNLISTKEY=NO or with ASSIGNLISTKEY=CREATE, the entry key is assigned the value specified for TARGETKEY.
  - If ASSIGNLISTKEY=MOVE or ASSIGNLISTKEY=ANY is specified, the entry key is assigned the list key value of the target list.
- When the request specifies ENTRYTYPE=ANY and the designated list entry does not exist, a new list entry will be created as follows:
  - If the structure was not allocated to use named entries or keyed entries, the newly created entry will be placed on the list specified by MOVETOLIST at the head or tail as specified by MOVETODIRECTION.
  - If the structure was allocated to use named entries, the entry name specified for ENTRYNAME is assigned to the newly created list entry, provided a list entry does not already exist with the same entry name. The newly created entry will be placed on the list specified by MOVETOLIST at the head or tail as specified by MOVETODIRECTION.
  - If the structure was allocated to use keyed entries, an entry key (and therefore, the target list keyed position) is assigned to the newly created list entry as follows:
    - If ENTRYKEY is not specified, and TARGETKEY=NO_TARGETKEY is specified (explicitly or by default) with ASSIGNLISTKEY=NO or with ASSIGNLISTKEY=MOVE, then:
      - If MOVETODIRECTION=HEADTOTAIL is specified (explicitly or by default), the newly created list entry is assigned an entry key value of all binary zeros.
      - If MOVETODIRECTION=TAILTOHEAD is specified, the newly created list entry is assigned an entry key value of all binary ones.
    - If ENTRYKEY is specified, and TARGETKEY=NO_TARGETKEY is specified (explicitly or by default) with ASSIGNLISTKEY=NO or with ASSIGNLISTKEY=MOVE, the newly created list entry is assigned the value specified by ENTRYKEY.
    - If TARGETKEY is specified with ASSIGNLISTKEY=NO or with ASSIGNLISTKEY=MOVE, the newly created list entry is assigned the value specified for TARGETKEY.
    - If ASSIGNLISTKEY=CREATE or ASSIGNLISTKEY=ANY is specified, the newly created list entry is assigned the list key value of the target list.
    - The newly created list entry is placed on the list specified by MOVETOLIST at the head or tail of the sublist composed of list entries whose entry keys are equal to the assigned entry key. The newly created list entry is placed at the head or tail of this sublist as specified by MOVETODIRECTION. If a sublist of entries with entry keys equal to the assigned entry key does not yet exist, the newly created list entry is placed on the list in key sequence.
    - The newly created list entry is placed on the list specified by MOVETOLIST at the head or tail of the sublist composed of list entries whose secondary keys are equal to the secondary key of the moved entry. The newly created list entry is placed at the head or tail of this sublist as specified by SKEYTARGETDIR. If a sublist of entries with secondary keys equal to the assigned secondary key does not yet exist, the newly created list entry is placed on the list in secondary key sequence.
  - If the structure was allocated to use user provided ENTRYIDs, the newly created list entry is assigned the ENTRYID specified by ASSIGNENTRYID, provided a list entry does not already exist with the same ENTRYID.

- If the structure supports entry data and BUFFER and BUFLIST are not specified, the newly created list entry will not contain any entry data.
- If the structure supports adjunct data and ADJAREA is not specified, the newly created list entry will contain binary zeros for adjunct data. If the structure was allocated to use secondary keys, then the first 32 bytes of the adjunct data will be occupied by the secondary key which may or may not be zeros.

DELETE
The designated list entry is to be removed from the list on which it resides and returned to the pool of free entries for reuse.

When the request completes successfully, the list entry controls for the deleted entry, the remaining number of entries or elements residing on the list, and the remaining total number of allocated entries in the structure are returned in the answer area specified by ANSAREA.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,SECONDARYKEY=NO_SECONDARYKEY
,SECONDARYKEY=*secondarykey*
Use this input parameter to specify an unsigned fixed 256-bit secondary key to be used to:

- Specify a secondary key to be used in conjunction with LISTNUM, DIRECTION, and SKEYREQTYPE to locate the list entry to be processed.
- Specify a secondary key to be assigned to the list entry if it is created. SKEYTARGETDIR is used to specify the position relative to secondary key ordering on the sublist at which to place the newly created list entry. If there exists a sublist of one or more entries with a matching secondary key on the list, the target position is at the head or tail of the sublist, as specified by SKEYTARGETDIR. If all existing list entries have a secondary key greater than that specified by SECONDARYKEY, the target position relative to secondary key ordering is at the head of the list. Similarly, if the specified SECONDARYKEY exceeds all of the secondary keys, the target position relative to secondary key ordering is at the tail of the list. If none of the list entries has a matching secondary key, and SECONDARYKEY is neither the greatest nor least among the secondary keys, the target position relative to secondary key ordering is determined according to secondary key sequence for the list.
- For structures allocated in a coupling facility of CFLEVEL=9 or higher, when the request specifies SKEYCOMPARE=YES, specifies the secondary key to be compared to the secondary key of the designated entry.

If the user does not supply SECONDARYKEY, the secondary key will be set to all binary zeros.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 32-byte field that contains the secondary key to be assigned to the list entry.

,SKEYCOMPARE=NO
,SKEYCOMPARE=YES
Use this input parameter to specify whether secondary key comparison should be performed to determine whether the list entry should be processed.

NO

No secondary key comparison should be performed to determine whether the list entry should be processed.

YES

Secondary key comparison should be performed to determine whether the list entry should be processed. if the designated list entry exists but the requested key comparison criterion is not met, the IXLLSTE request is terminated.

SKEYCOMPARE=YES is only meaningful when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

SKEYCOMPARE=YES will be ignored if the target structure was not allocated with secondary keys.

,SKEYPOSITION=UPDATE
,SKEYPOSITION=KEEP

Use this input parameter to specify whether the list entry is moved from its current position on the secondary sublist when REQUEST=MOVE. This keyword has effect only if the list number specified by MOVETOLIST is the same as the list number on which the list entry currently resides.

The SKEYPOSITION keyword is only meaningful for list structures allocated in a coupling facility of CFLEVEL=9 or higher.

UPDATE

The list entry should be moved from its current position on the secondary sublist to a position on the secondary sublist as specified by SKEYTARGETDIR.

KEEP

The list entry should keep its current position based on secondary key ordering on the secondary sublist if and only if the list number specified by MOVETOLIST matches the current list number that contains the list entry.

,SKEYREQTYPE=EQUAL
,SKEYREQTYPE=LESSOREQUAL
,SKEYREQTYPE=GREATEROREQUAL

Use this input parameter to specify how secondary key comparison is to be performed on the designated keyed list entry or, when LOCATOR=KEYPOS is specified, to designate the list entry to be processed.

EQUAL

- The designated list entry must have a secondary key equal to the value specified for SECONDARYKEY. If the secondary key of the entry is not equal to the value specified for SECONDARYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that if a list entry exists with a secondary key equal to the value specified for SECONDARYKEY, that entry is designated for processing. If more than one such entry exists, the list entry at either the head or tail of the sublist as specified by DIRECTION is designated for processing. If no such entry exists, the IXLLSTE request is terminated.

LESSOREQUAL

- The designated list entry msut have a secondary key equal to or less than the value specified for SECONDARYKEY. If the secondary key of the entry is not equal to or less than the value specified for SECONDARYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that:
  – If a list entry exists with a secondary key equal to or less than the value specified for SECONDARY key, that entry is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.

- If no list entries exist with an equal secondary key, the list entry with the highest secondary key less than the value specified for SECONDARYKEY is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
- If no list entries exist with a secondary key equal to or less than the value specified for SECONDARYKEY, the IXLLSTE request is terminated.

GREATEROREQUAL

- The designated list entry must have a secondary key equal to or greater than the value specified for SECONDARYKEY. If the secondary key of the entry is not equal to or greater than the value specified for SECONDARYKEY, the IXLLSTE request is terminated.
- When LOCATOR=KEYPOS, specifies that:
  - If a list entry exists with a secondary key equal to the value specified for SECONDARYKEY, that entry is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  - If no list entries exist with an equal secondary key, the list entry with the lowest secondary key greater than the value specified for SECONARYKEY is designated for processing. If more than one such entry exists, the list entry at the head or tail of the sublist as specified by DIRECTION is designated for processing.
  - If no list entries exist with a secondary key equal to or greater than the value specified for SECONDARYKEY, the IXLLSTE request is terminated.

,SKEYTARGETDIR=HEADTOTAIL
,SKEYTARGETDIR=TAILTOHEAD

Use this input parameter to specify the position relative to secondary key ordering on the sublist at which to place the newly created list entry. This parameter is used in conjunction with LISTNUM and with the assigned secondary key value when the structure was allocated with secondary keys.

SKEYTARGETDIR is only valid when the structure is allocated in a coupling facility of CFLEVEL=9 or higher.

HEADTOTAIL

The designated position is at the head of the sublist designated by the secondary key value.

TAILTOHEAD

The designated position is at the tail of the sublist designated by the secondary key value.

,TARGETKEY=NO_TARGETKEY
,TARGETKEY=*targetkey*

Use this input parameter to specify an unsigned fixed 128-bit entry key to be assigned to the list entry if it is moved or created as a result of this request. The assigned entry key is used in conjunction with MOVETOLIST and MOVETODIRECTION to designate the target keyed position of the list entry.

This keyword is only applicable to a structure that was allocated to used keyed entries, and is otherwise ignored.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte area that contains the entry key to be assigned to the list entry.

,UPDATECURSOR=NO
,UPDATECURSOR=YES

Use this input parameter to specify whether the list cursor for the list containing the processed list entry should be updated if the request is successful.

If the structure was allocated with keyed entries, the entry key order is always used to determine the next or previous list entry. Note that the secondary key order is not relevant in determining the next or previous list entry.

NO

The cursor should not be updated.

Note that the list cursor will be set to binary zeros if it points to an entry that is deleted or moved to another list as a result of this request regardless of how UPDATECURSOR is specified.

YES
The cursor should be updated.

If the list entry specified on the request exists, the list cursor for the list on which the entry resides is updated before the list entry is processed for the request.

The CURSORUPDTYPE values of NEXTCOND, CURRENT, and CURRENTCOND are only meaningful for list structures allocated in a coupling facility of CFLEVEL=1 or higher.

,VERSCOMP=NO_VERSCOMP
,VERSCOMP=*verscomp*
Use this input parameter to specify a version number to be compared to the version number of the existing entry. If this request creates a new entry, the VERSCOMP specification is ignored.

The existing entry is moved only if its version number meets the condition specified by the VERSCOMPTYPE parameter. If the entry does not have the specified version number, the request is terminated with no change to the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPARE=NO
,VERSCOMPARE=YES
Use this input parameter to specify whether version number comparison should be performed to determine whether the list entry should be processed.

NO
No version number comparison should be performed to determine whether the list entry should be processed.

YES
Version number comparison should be performed to determine whether the list entry should be processed.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL
Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

,VERSUPDATE=NONE
,VERSUPDATE=INC
,VERSUPDATE=DEC
,VERSUPDATE=SET
Use this input parameter to specify how the entry version number of the moved entry will be updated, or for those cases where a new entry is created, initialized.

NONE
The existing entry's version number is not updated. The new entry's version number is set to binary zeros.

INC
The existing entry's version number is increased by one. The new entry's version number is set to binary zeros, except for the low-order bit, which is set to one.

DEC
The existing entry's version number is decreased by one. The new entry's version number is set to all binary ones.

SET
Both the existing and the new entry's version number is set to the value specified by NEWVERS.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 73. Return and Reason Codes for IXLLSTE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The IXLLSTE READ, MOVE, or DELETE request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved. If supported and requested, the entry data was retrieved. If this was a MOVE request, the entry was moved. If this was a DELETE request, the entry was deleted. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLSTE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTE macro. • If LAALCTL is not zeros, the request completed prematurely. See the action suggested for return code X'4' reason code X'xxxx0409'. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** For a LOCKOPER=HELDBY or LOCKMODE=COND request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLSTE while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLSTE macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0837	**Equate Symbol:** IXLRSNCODEBADWRITEADJDATA **Meaning:** Program error. The IXLLSTE WRITE or MOVE request specified that adjunct data was to be written, but the area specified by ADJAREA is not addressable. No entry data was written. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTE while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTE in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLSTE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTE macro.
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTE while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTE macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLSTE while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLSTE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTE macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx083E	**Equate Symbol:** IXLRSNCODEMAXLISTKEY **Meaning:** Program error. The list key to be assigned to an entry that was being created or moved was not valid. Either the list key or the list key plus the list key increment value was greater than the maximum list key. **Action:** Ensure that you specified a correct list key increment. Depending on the protocol you are using for assigning list key values, you might want to issue a WRITE_LCONTROLS request to update either the list key value to a lower value or the maximum list key value to a higher value.
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The entry designated by the specified list entry controls has a version number that does not meet the criteria specified by VERSCOMP and VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the values specified in VERSCOMP and VERSCOMPTYPE and look at the version returned in the answer area.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by the specified list entry controls does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0841	**Equate Symbol:** IXLRSNCODEBADENTRYNAME **Meaning:** Program error. The name specified by ENTRYNAME is not a unique name within the structure, and therefore entry creation is suppressed. **Action:** Be sure to provide a unique entry name when creating entries to be written to structures that support entry names.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying an unconditional SET or NOTHELD lock operation failed because the lock was held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0848	**Equate Symbol:** IXLRSNCODEBADRESET **Meaning:** Program error. LOCKOPER=RESET was specified for a lock not currently held by the invoker. The value of the connection ID holding the lock is returned in the answer area (field LAACONID). **Action:** Check your code to ensure that your connection did not already reset the lock.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLSTE request type either required the structure to support entry keys or designated a sublist, list entry, or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx084E	**Equate Symbol:** IXLRSNCODEBADMOVETOLIST **Meaning:** Program error. The list number specified for MOVETOLIST exceeds the number of lists defined for the structure. **Action:** Correct MOVETOLIST to specify a list that exists in the structure. The maximum number of lists in a structure is determined by the LISTHEADERS parameter on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority value for the specified list does not meet the criteria specified by AUTHCOMP and AUTHCOMPTYPE. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLSTE is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data for the first entry to be read in the list. No data is returned. **Action:** If more space is available, specify a larger buffer size, or more buffers, and reissue the request. Check the information returned in the answer area (mapped by IXLYLAA) in the field LAALCTL (mapped by IXLYLCTL).

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect. For requests that write coupling facility data, the data cannot be fetched from the specified buffer area. For requests that read coupling facility data, the data cannot be stored into the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLSTE in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTE macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The value specified for ELEMNUM is not valid. **Action:** Correct the ELEMNUM specification to be within the allowable range: from zero to whatever MAXELEMNUM value was returned to the connector in the connect answer area.
8	xxxx0890	**Equate Symbol:** IXLRSNCODEBADENTRYIDVALUE **Meaning:** Program error. The specified user entry ID is zero. **Action:** Correct the value of the user entry ID.
8	xxxx0894	**Equate Symbol:** IXLRSNCODEBADKEYCOMPARE **Meaning:** Program error. The specified key comparison criteria was not satisfied. **Action:** Ensure that the values specified for either ENTRYKEY or SECONDARYKEY are valid.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0896	**Equate Symbol:** IXLRSNCODEDUPLICATEENTRYID **Meaning:** Program error. The specified user entry ID already exists in the specified structure. **Action:** Ensure that the value specified for the entry ID is valid.
8	xxxx0897	**Equate Symbol:** IXLRSNCODEBADKEYTYPE **Meaning:** Program error. The specified KEYTYPE value is not valid for the specified structure. **Action:** Ensure that the value of KEYTYPE is valid.
8	xxxx0899	**Equate Symbol:** IXLRSNCODEBADSKEYCOMPARE **Meaning:** Program error. The specified SKEYCOMPARE value is not valid for the specified structure. **Action:** Correct the value of SKEYCOMPARE.
8	xxxx089A	**Equate Symbol:** IXLRSNCODEBADSKEYREQTYPE **Meaning:** Program error. The specified SKEYREQTYPE value is not valid for the specified structure. **Action:** Correct the value of SKEYREQTYPE.
8	xxxx089B	**Equate Symbol:** IXLRSNCODEBADKEYCOMPARETYPE **Meaning:** Program error. The specified KEYCOMPARE value is not valid for the specified structure. **Action:** Correct the value of KEYCOMPARE.
8	xxxx089C	**Equate Symbol:** IXLRSNCODEBADMOVETOKEY **Meaning:** Program error. The specified MOVETOKEY value is not valid for the specifed structure. **Action:** Ensure that the value of MOVETOKEY is correct.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C17	**Equate Symbol:** IXLRSNCODESTRFULL **Meaning:** Environmental error. The request attempted to create a new entry or overwrite an existing entry, but the structure is full and cannot accommodate any more entries. **Action:** Determine why the structure is full. You should be monitoring the usage of the structure every time a read or write is done (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure). This data should be evaluated periodically to ensure structure resources are being used efficiently. You might be able to delete existing entries to free up space, rebuild the structure to make it larger if rebuild is allowed (IXLCONN macro, ALLOWREBLD parameter), or alter the size of the structure (IXLALTER macro).
C	xxxx0C18	**Equate Symbol:** IXLRSNCODELISTFULL **Meaning:** Environmental error. The list designated as the target of a MOVE request, or as the target for entry createion on a WRITE or MOVE request, cannot accomodate more entries. **Action:** If appropriate, change the maximum number of entries allowed on the list by specifying a new LISTLIMIT on a WRITE_LCONTROLS request. You should be monitoring the usage of the list structure every time a read or write is done. (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure.) This data should be evaluated periodically to ensure structure resources are being used efficiently.

Table 73. Return and Reason Codes for IXLLSTE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLLSTM — XES List Structure Multiple Entry Services

Description

The IXLLSTM service enables you to perform operations to read, move, or delete multiple list entries. Some functions of IXLLSTM require that the list structure be allocated in a coupling facility of CFLEVEL=9 or higher. The following services are available:

- Delete multiple list entries that are designated in a list contained in the storage area specified by BUFFER or BUFLIST.
- Delete list entries that meet a designated criteria from a list.
- Delete multiple list entries from a coupling facility list structure.
- Move multiple list entries that are designated in a list contained in the storage area specified by BUFFER or BUFLIST from the current location to a target location.
- Read multiple entries from a single list into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA and ANSAREA.
- Read multiple entries from the list structure into the storage areas specified by BUFFER or BUFLIST and/or ADJAREA and ANSAREA.

The IXLLSTM macro provides list services equivalent to the following IXLLIST request types:
- IXLLIST REQUEST=DELETE_ENTRYLIST
- IXLLIST REQUEST=DELETE_MULT
- IXLLIST REQUEST=READ_LIST
- IXLLIST REQUEST=READ_MULT

See the descriptions of the corresponding IXLLIST macro for basic information about the services provided.

List structures allocated in a coupling facility of CFLEVEL=9 or higher can be allocated with certain attributes not available in a lower level coupling facility.
- Users can assign a user-designated entry identifier to a newly created list entry, rather than having the system assign the entry identifier.
- Secondary keys can be specified, thus allowing the user to reference a keyed list entry by entry key, secondary key, or both.

The IXLLSTM macro can be used to exploit these new attributes.

See *z/OS MVS Programming: Sysplex Services Guide* for information about using the IXLLSTM macro.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The current primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued for the structure.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	Disabled callers must be legally disabled by holding the CPU lock and cannot hold other disabled locks. Enabled callers must not hold any locks. When MODE=SYNCSUSPEND is specified, the caller must be enabled.
Control parameters:	See “Restrictions” on page 1116

Programming Requirements

- If your program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLLSTM. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.
- Include the IXLYCON mapping macro in your program. This macro provides a list of equate symbols for users of XES services and exits.
- Include mapping macros IXLYLAA, IXLYLCTL, IXLYDELI, and IXLYMELI in your program as necessary. Table 74 lists these macros, the information and areas they map, and the particular IXLLIST requests and parameters they apply to.

Table 74. Mapping Macros for IXLLSTM

Mapping Macro	**Information**	**Area**	**Request/Parameter**
IXLYLAA	Answer area output	ANSAREA area	All requests
IXLYLCTL	List entry controls	Field LAALCTL of IXLYLAA	READ_MULT
		Field LAARLRMLCTLS of IXLYLAA, BUFLIST buffers, BUFFER area	READ_LIST & READ_MULT (when TYPE=ECONTROLS)
IXLYDELI	Delete Entry List input	BUFLIST buffers, BUFFER area	DELETE_ENTRYLIST
IXLYMELI	Move Entry List input	BUFLIST buffers, BUFFER area	MOVE_ENTRYLIST

Restrictions

- If you specify BUFLIST and PAGEABLE=YES, all of the buffers in the list must be in the same area of storage; you cannot mix common and private storage addresses for the buffers in the list.
- This service cannot be invoked by callers running as a disabled interrupt exit (DIE).
- The caller's parameter list must be addressable in the caller's primary address space.
- If the caller is running in AR ASC mode and specifies a parameter using explicit register notation, the access register corresponding to the general register must appropriately qualify the general register.
- The virtual storage areas specified by the ADJAREA and ANSAREA parameters must be addressable in the caller's primary address space or from the caller's PASN access list.
- The virtual storage areas specified by parameters other than ADJAREA and ANSAREA can be addressable in the caller's primary, secondary, or home address spaces, from the PASN access list, or from the dispatchable unit access list (DU-AL).
- If the caller is disabled then the parameter list and all storage areas addressed by the macro parameters must reside in either nonpageable or disabled reference storage.

Input Register Information

Before issuing the IXLLSTM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0	Reason code
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system

15	Return code

When control returns to the caller, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Performance Implications

Please see *z/OS MVS Programming: Sysplex Services Guide* for performance information.

Understanding IXLLSTM Version Support

The IXLLSTM macro supports versions 0, 1, 3, and 4 — the version number corresponds to the level of the IXLLIST or IXLLSTM macro in which a keyword or function was introduced. The higher the version number, the more recent the version of the macro. Some IXLLSTM keywords are supported for multiple versions, but have different functions for each version.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLLSTM macro.
- The following IXLLSTM keywords are supported by all versions starting with version 1 and higher of the IXLLSTM macro:

KEYCOMPARE	LISTKEYINC

- The following keyword is supported by all versions starting with version 3 and higher of the IXLLSTM macro:

EXTRESTOKEN

- The following keywords and functions are supported by all versions starting with version 4 and higher of the IXLLSTM macro:

LISTKEYAREA	SECONDARYKEY	SKEYRANGEEND
MISCOMPARE	SKEYCOMPARE	SKEYREQTYPE
MOVETOSKEY		

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Summary of Version-Dependent Parameter Functions

The following table summarizes the allowed use of some of the specifications that can be made depending on the PLISTVER value specified.

Table 75. IXLLSTM Version Support

Keyword	Version	Notes
AUTHCOMPARE	0	NO may be specified.
	1	NO or YES may be specified.
REQUEST	0	READ_LIST, READ_MULT, DELETE_MULT, or DELETE_ENTRYLIST may be specified.
	1	Same as version 0
	2	Same as version 1
	3	Same as version 2

Table 75. IXLLSTM Version Support (continued)

Keyword	Version	Notes
	4	READ_LIST, READ_MULT, DELETE_MULT, DELETE_ENTRYLIST, DELETE_LIST, or MOVE_ENTRYLIST may be specified.
VERSCOMPARE	0	NO or YES may be specified.
	1	Same as version 0
	2	Same as version 1
	3	Same as version 2
	4	NO, YES, or BYENTRY may be specified.
VERSCOMPTYPE	0	EQUAL may be specified.
	1	EQUAL ro LESSOREQUAL may be specified.
KEYSCANTYPE	0	ENTRY may be specified.
	1	Same as version 0.
	2	Same as version 1.
	3	Same as version 2.
	4	ENTRY or SECONDARY may be specified.
KEYREQTYPE	0	EQUAL may be specified. GREATEROREQUAL and LESSOREQUAL may be specified for only REQUEST=READ_LIST.
	1	Same as version 0
	2	Same as version 1
	3	Same as version 2
	4	EQUAL, LESSOREQUAL, GREATEROREQUAL, or RANGE may be specified.

Syntax Diagram

The syntax diagram for IXLLSTM is as follows:

main diagram

```
                                    ┌─,REQID=NO_REQID─┐
►►──IXLLSTM──b──CONTOKEN=contoken───┼─────────────────┼──────────────────────────────────────────────►
                                    └─,REQID=reqid────┘

►─,REQUEST=─┬─READ_LIST──┤ parameters-1 ├──────────┬──────────────────────────────────────────────────►
            ├─DELETE_LIST──┤ parameters-2 ├────────┤
            ├─READ_MULT──┤ parameters-3 ├──────────┤
            ├─DELETE_MULT──┤ parameters-4 ├────────┤
            ├─MOVE_ENTRYLIST──┤ parameters-5 ├─────┤
            └─DELETE_ENTRYLIST──┤ parameters-6 ├───┘


  ┌─,LOCKINDEX=NO_LOCKINDEX─────────────────────────────────────────┐
►─┼─────────────────────────────────────────────────────────────────┼────────────────────────────────►
  └─,LOCKINDEX=lockindex─,LOCKOPER=─┬─NOTHELD───────────────────────┤
                                    │        ┌─,LOCKCOMP=NO_LOCKCOMP─┐ │
                                    └─HELDBY─┼───────────────────────┼─┘
                                             └─,LOCKCOMP=lockcomp────┘

  ┌─,MODE=SYNCSUSPEND─,ANSAREA=ansarea─,ANSLEN=anslen──────────────────┐
►─┼────────────────────────────────────────────────────────────────────┼──┬──────────────────┬──────►
  ├─,MODE=SYNCECB─,REQECB=reqecb─,ANSAREA=ansarea─,ANSLEN=anslen───────┤  └─,RETCODE=retcode─┘
  │                ┌─,REQDATA=NO_REQDATA─┐                             │
  ├─,MODE=SYNCEXIT─┼─────────────────────┼─,ANSAREA=ansarea─,ANSLEN=anslen─┤
  │                └─,REQDATA=reqdata────┘                             │
  ├─,MODE=SYNCTOKEN─,REQTOKEN=reqtoken─,ANSAREA=ansarea─,ANSLEN=anslen─┤
  ├─,MODE=ASYNCECB─,REQECB=reqecb─,ANSAREA=ansarea─,ANSLEN=anslen──────┤
  │                 ┌─,REQDATA=NO_REQDATA─┐                            │
  ├─,MODE=ASYNCEXIT─┼─────────────────────┼─,ANSAREA=ansarea─,ANSLEN=anslen─┤
  │                 └─,REQDATA=reqdata────┘                            │
  ├─,MODE=ASYNCTOKEN─,REQTOKEN=reqtoken─,ANSAREA=ansarea─,ANSLEN=anslen┤
  └─,MODE=ASYNCNORESPONSE──────────────────────────────────────────────┘

                       ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────┐
►─┬──────────────────┬─┼───────────────────────────┼──┼────────────────────────────────────┼──────►◄
  └─,RSNCODE=rsncode─┘ ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D──────┐      │
                       └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl───┼──────────┼─)────┤
                                                      │                  └─,mfattr──┘      │
                                                      │                  ┌─,COMPLETE─┐     │
                                                      └─,MF=(E─,mfctrl───┼───────────┼─)───┘
                                                                         └─,COMPLETE─┘
```

parameters-1

```
                         ┌─,ADJAREA=NO_ADJAREA─┐  ┌─,DIRECTION=HEADTOTAIL─┐
►►──,TYPE=──┬─ENTDATA───┬─┴─────────────────────┴──┴───────────────────────┴──────────►
            ├─ADJDATA───┤ └─,ADJAREA=adjarea────┘  └─,DIRECTION=TAILTOHEAD─┘
            └─ECONTROLS─┘

►─,LOCATOR=──┬─CURSOR──┤ parameters-15 ├───┬──────────────────────────────────────────►
             ├─ENTRYID─┤ parameters-16 ├───┤
             ├─ENTRYNAME─┤ parameters-17 ├─┤
             ├─UNKEYPOS─┤ parameters-15 ├──┤
             └─KEYPOS──┤ parameters-18 ├───┘

  ┌─,VERSCOMPARE=NO──────────────────────────────────────────────────┐
►─┴──────────────────────────────────────────────────────────────────┴─┤ parameters-8 ├──────►◄
  │                                     ┌─,VERSCOMPTYPE=EQUAL──────┐ │
  └─,VERSCOMPARE=YES─,VERSCOMP=verscomp─┴──────────────────────────┴─┘
                                        └─,VERSCOMPTYPE=LESSOREQUAL┘
```

parameters-2

```
  ┌─,DIRECTION=HEADTOTAIL─┐
►►┴───────────────────────┴─,LOCATOR=──┬─CURSOR──┤ parameters-15 ├───┬──────────────────►◄
  └─,DIRECTION=TAILTOHEAD─┘            ├─ENTRYID─┤ parameters-16 ├───┤
                                       ├─ENTRYNAME─┤ parameters-17 ├─┤
                                       ├─UNKEYPOS─┤ parameters-15 ├──┤
                                       └─KEYPOS──┤ parameters-18 ├───┘
```

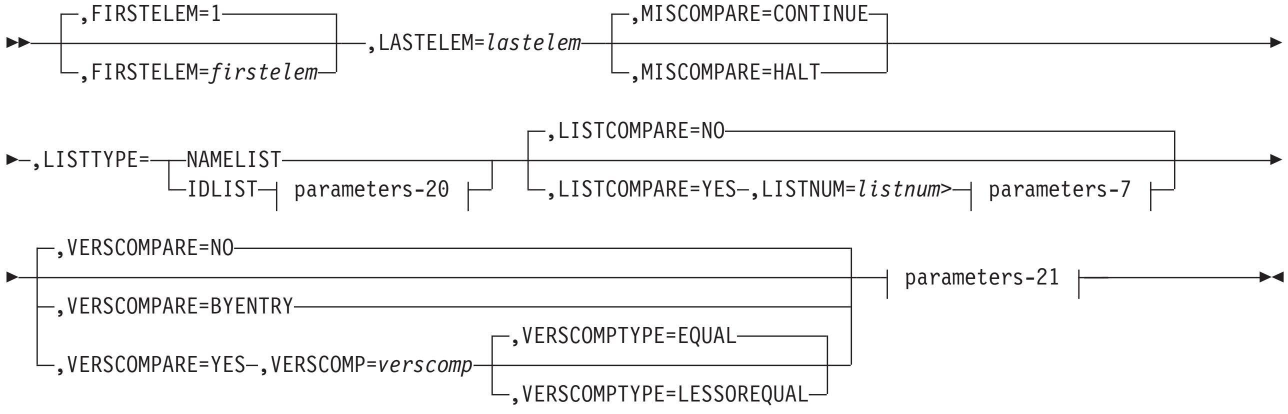
parameters-3

```
                         ┌─,ADJAREA=NO_ADJAREA─┐
►►──,TYPE=──┬─ENTDATA───┬─┴─────────────────────┴─┤ parameters-4 ├──┤ parameters-8 ├──────►◄
            ├─ADJDATA───┤ └─,ADJAREA=adjarea────┘
            └─ECONTROLS─┘
```

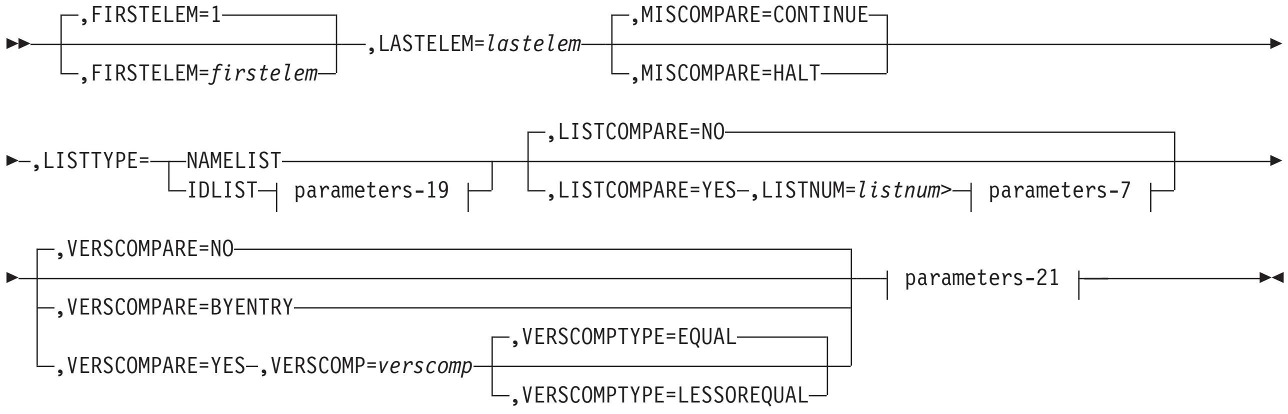
parameters-4

```
     ┌─,RESTOKEN=NO_RESTOKEN─┐        ┌─,LISTCOMPARE=NO───────────────────────────────────────┐
►►─┬─┴───────────────────────┴─────┬──┴───────────────────────────────────────────────────────┴──►
   │ └─,RESTOKEN=restoken────┘     │  └─,LISTCOMPARE=YES─,LISTNUM=listnum─┤ parameters-7 ├───┘
   │ ┌─,EXTRESTOKEN=NO_EXTRESTOKEN─┤
   └─┴─────────────────────────────┘
     └─,EXTRESTOKEN=extrestoken────┘

  ┌─,KEYCOMPARE=NO──────────────────────┐  ┌─,SKEYCOMPARE=NO──────────────────────┐
►─┴─────────────────────────────────────┴──┴──────────────────────────────────────┴──────────►
  └─,KEYCOMPARE=YES─┤ parameters-13 ├───┘  └─,SKEYCOMPARE=YES─┤ parameters-14 ├───┘

  ┌─,VERSCOMPARE=NO──────────────────────────────────────────────────┐
►─┴──────────────────────────────────────────────────────────────────┴────────────────────►◄
  │                                     ┌─,VERSCOMPTYPE=EQUAL──────┐ │
  └─,VERSCOMPARE=YES─,VERSCOMP=verscomp─┴──────────────────────────┴─┘
                                        └─,VERSCOMPTYPE=LESSOREQUAL┘
```

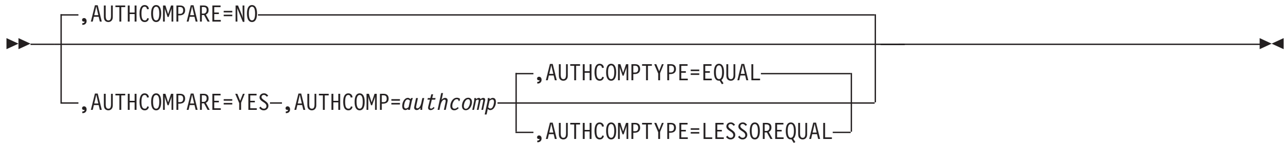
parameters-5



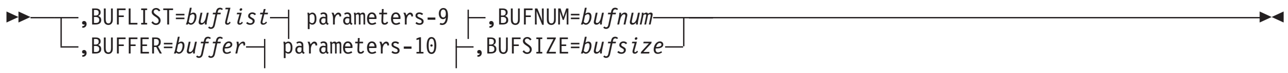
parameters-6



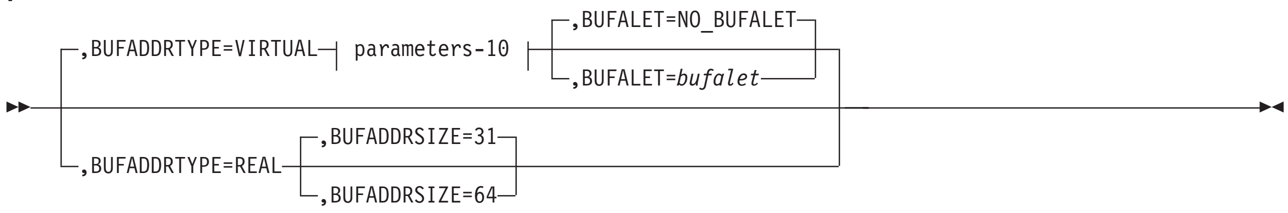
parameters-7



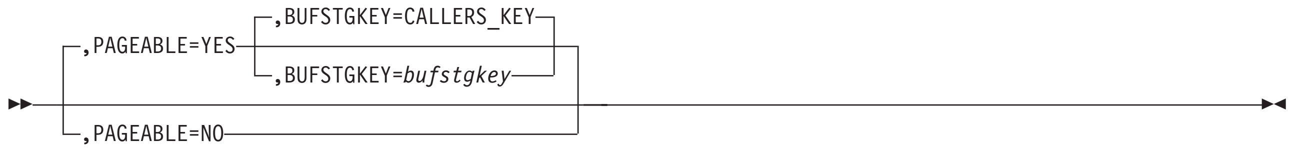
parameters-8



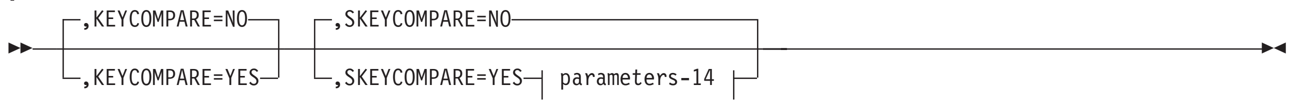
parameters-9



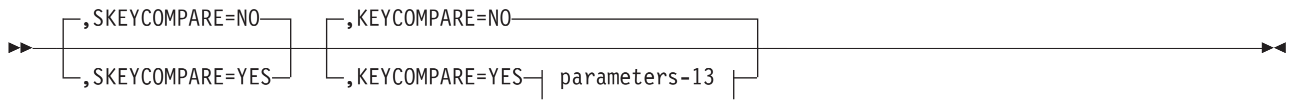
parameters-10



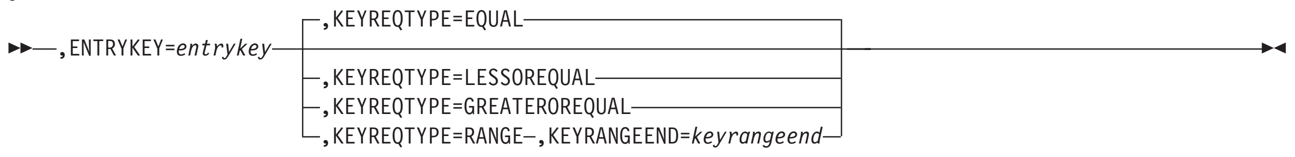
parameters-11



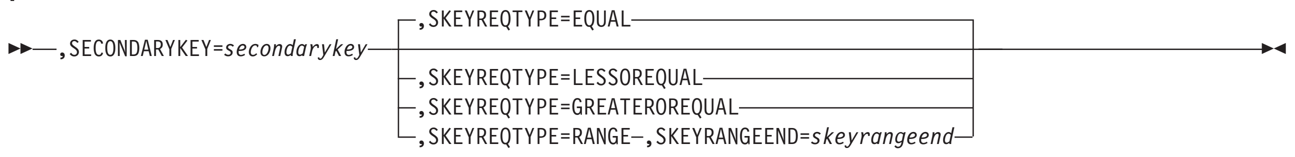
parameters-12



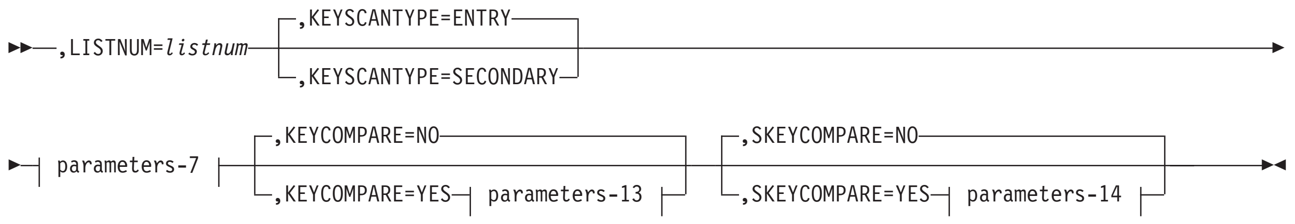
parameters-13



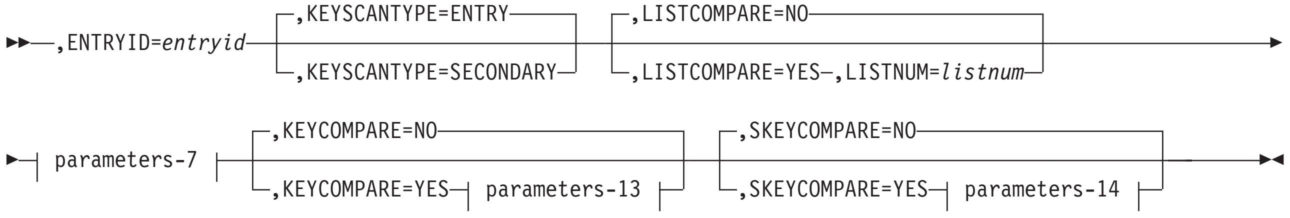
parameters-14


,SECONDARYKEY=*secondarykey*
,SKEYREQTYPE=EQUAL
,SKEYREQTYPE=LESSOREQUAL
,SKEYREQTYPE=GREATEROREQUAL
,SKEYREQTYPE=RANGE
,SKEYRANGEEND=*skeyrangeend*

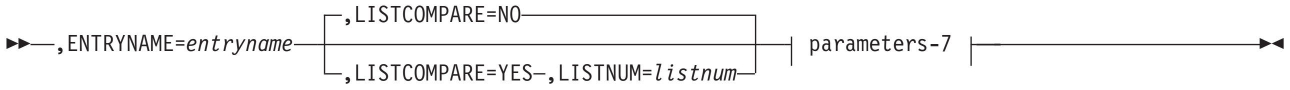
parameters-15



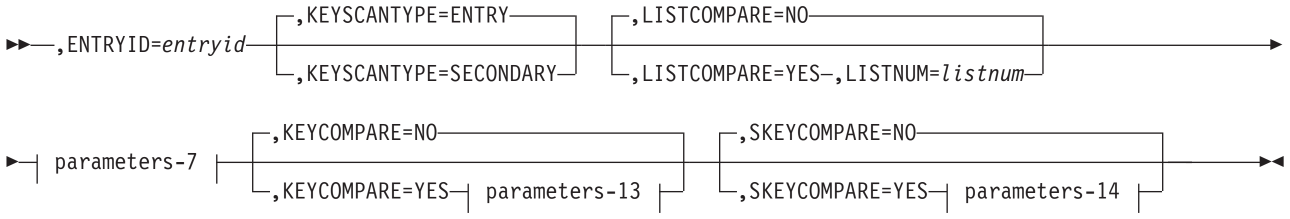
parameters-16



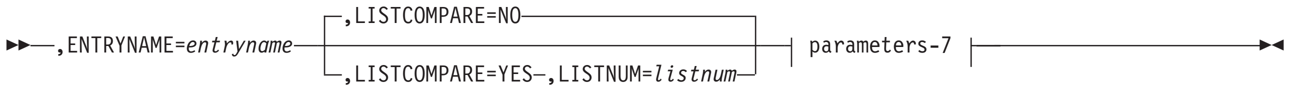
parameters-17



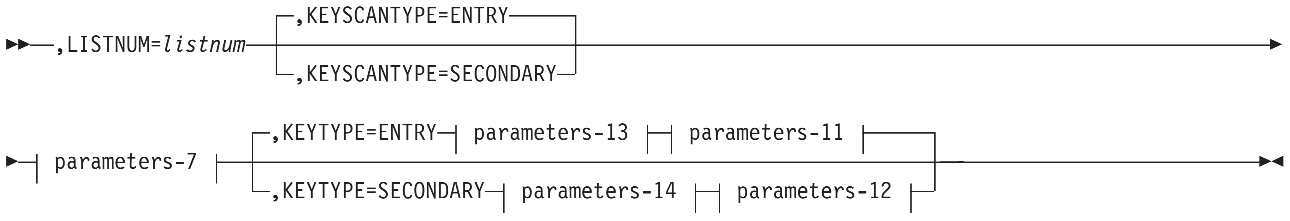
parameters-16



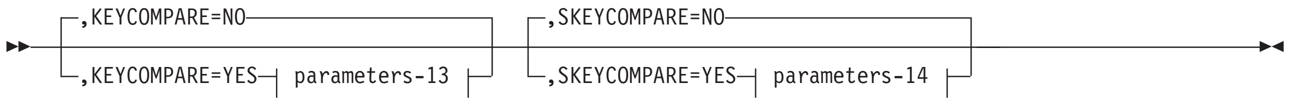
parameters-17



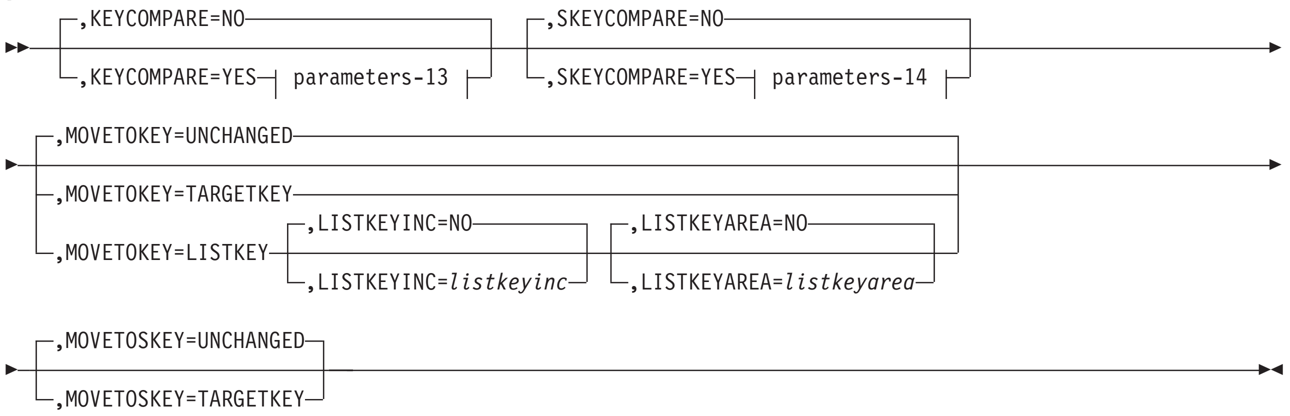
parameters-18



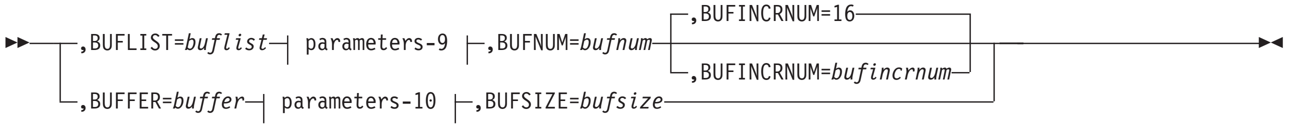
parameters-19



parameters-20



parameters-21



Parameter Descriptions

The parameter descriptions for IXLLSTM are listed in alphabetical order. Default values are underlined:

,ADJAREA=<u>NO_ADJAREA</u>
,ADJAREA=*adjarea*

Use this input parameter to specify a storage area to contain the adjunct data that is read from or written to an entry.

Specify ADJAREA only for structures that support adjunct data. (Adjunct areas for a structure are established through the IXLCONN macro.)

If the structure was allocated to use secondary keys, the first 32 bytes of ADJDATA will contain the secondary key of the entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-byte area that contains or will contain the adjunct data.

,ANSAREA=*ansarea*

Use this input parameter to specify an answer area to contain information returned from the request. The format of the answer area is described by mapping macro IXLYLAA.

Not all fields in the answer area are applicable to all request types. Request type descriptions indicate which answer area fields are applicable for successful request completion cases. Return and reason code description indicates which answer area fields are applicable for non-successful completing requests.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of ANSLEN) that will contain the information returned by the request.

,ANSLEN=*anslen*

Use this input parameter to specify the size of the storage area specified by ANSAREA.

Check the prologue of the IXLYLAA mapping macro for the minimum required size of the answer area, or use the length field (LAA_LEN) in the LAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 2-byte field that contains the size, in bytes, of the answer area.

,AUTHCOMP=NO_AUTHCOMP
,AUTHCOMP=*authcomp*

Use this input parameter to specify a value to be compared to the list authority value of the list specified by LISTNUM. You must supply the LISTNUM parameter.

If the comparison does not meet the condition specified by the AUTHCOMPTYPE parameter (EQUAL or LESSOREQUAL), the request fails.

Note: The AUTHCOMP parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-byte field that contains the list authority value.

,AUTHCOMPARE=NO
,AUTHCOMPARE=YES

Use this input parameter to specify whether the list authority comparison is to be used to determine if entries on the list should be processed.

NO
No list authority comparison is to be performed before processing any of the list entries.

YES
List authority comparison should precede processing of any list entries.

,AUTHCOMPTYPE=EQUAL
,AUTHCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how the list authority comparison is to be performed.

EQUAL
The list authority for the list specified by LISTNUM must be equal to the value specified for AUTHCOMP.

LESSOREQUAL
The list authority for the list specified by LISTNUM must be less than or equal to the value specified for AUTHCOMP.

Note: The AUTHCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

,BUFADDRSIZE=31
,BUFADDRSIZE=64

Use this input parameter to specify whether a 31-bit or a 64-bit address is specified by a BUFLIST entry.

31 The entry in BUFLIST is 31 bits in size.

64 The entry in BUFLIST is 64 bits in size.

,BUFADDRTYPE=VIRTUAL
,BUFADDRTYPE=REAL

Use this input parameter to specify whether the buffer addresses specified in the BUFLIST list are virtual storage or real storage addresses.

VIRTUAL
The buffer addresses are virtual storage addresses. The virtual storage can be pageable or nonpageable. See the PAGEABLE parameter for information about managing storage binds when specifying virtual storage addresses.

REAL
The buffer addresses are real storage addresses.

It is the caller’s responsibility to manage the binds between the data buffer virtual storage and the real storage addresses provided. The caller must ensure that the data buffer virtual storage remains bound to the real storage addresses provided until the request completes.

,BUFALET=NO_BUFALET
,BUFALET=*bufalet*
Use this input parameter to specify an access list entry token (ALET) to be used in referencing all of the buffers specified by BUFLIST.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the ALET.

,BUFFER=*buffer*
Use this parameter to hold data for the request. The BUFSIZE keyword specifies the size of the buffer. For READ_LIST and READ_MULT requests, BUFFER is an output parameter. For MOVE_ENTRYLIST and DELETE_ENTRYLIST requests, BUFFER is an input parameter. For all IXLLSTM request types, the length of the buffer must be a multiple of 4096 bytes between 4096 and 65536 and the buffer must start on a 4096-byte (page) boundary.

- Upon successful completion of a READ_LIST or READ_MULT request, the BUFFER area is used for output and contains, starting at offset zero, an array of elements. One array element is returned for each processed entry. The number of elements returned in the BUFFER area is indicated in the answer area. The length of an array element can be determined by its make-up: the structure element size, the number of elements in the entry, an adjunct data size (64 bytes), and the length of list entry controls. (The length and contents of list entry controls is defined by mapping macro IXLYLCTL.)

  For READ_LIST and READ_MULT requests, each array element is constructed as follows, dependent on the request options specified:

  – When adjunct data is requested, the adjunct data for the first entry processed is returned in the storage area specified by ADJAREA. The adjunct data for all other entries processed is returned in the BUFFER area.

  – When list entry controls are requested, the entry controls for the first entry processed are returned in the answer area specified by ANSAREA. The entry controls for all other entries processed are returned in the BUFFER area.

  The format of each array element in the BUFFER, therefore, is as follows:

  – When TYPE=ENTDATA is specified, entry data for each list entry processed is contained in the buffer.

  – When TYPE=ADJDATA is specified, adjunct data for each list entry processed after the first entry is contained in the buffer. (The adjunct data for the first entry processed is returned in ADJAREA.)

  – When TYPE=ECONTROLS is specified, list entry controls for each list entry processed after the first entry is contained in the buffer. (The list entry controls for the first entry is returned in ANSAREA.)

  – When TYPE=(ENTDATA,ADJDATA) is specified, entry data for the first list entry processed is contained in the buffer, followed by the entry data and then the adjunct data for each additional list entry processed.

  – When TYPE=(ENTDATA,ECONTROLS) is specified, entry data for the first list entry processed is contained in the buffer, followed by the list entry controls and then the entry data for each additional list entry processed.

  – When TYPE=(ADJDATA,ECONTROLS) is specified, list entry controls followed by adjunct data for each list entry processed after the first is contained in the buffer.

  – When TYPE=(ENTDATA,ADJDATA,ECONTROLS) is specified, entry data for the first list entry processed is contained in the buffer, followed by list entry controls, entry data, and adjunct data (in that order) for each additional list entry processed.

- For MOVE_ENTRYLIST requests, BUFFER is used for input and should be formatted into 32-byte, 64-byte, or 96-byte elements, where each element is mapped by IXLYMELI and contains the information required to move a list entry. The format and size of an element is determined by the options specified on the MOVE_ENTRYLIST request.
  - A 32-byte element is required for any one of the following conditions:
    - The structure does not support keyed entries and VERSCOMPARE=NO is specified.
    - The structure does not support keyed entries and VERSCOMPARE=YES is specified.
    - The structure does support keyed entries and MOVETOKEY=UNCHANGED, MOVETOSKEY=UNCHANGED, and VERSCOMPARE=NO are specified.
    - The structure does support keyed entries and MOVETOKEY=UNCHANGED, MOVETOSKEY=UNCHANGED, and VERSCOMPARE=YES are specified.
    - The structure does support keyed entries and MOVETOKEY=LISTKEY, MOVETOSKEY=UNCHANGED, and VERSCOMPARE=NO are specified.
    - The structure does support keyed entries and MOVETOKEY=LISTKEY, MOVETOSKEY=UNCHANGED, and VERSCOMPARE=YES are specified.
  - A 64-byte element is required for the following conditions:
    - VERSCOMPARE=BYENTRY or MOVETOKEY=TARGETKEY is specified.
  - A 96-byte element is required for the following condition:
    - MOVETOSKEY=TARGETKEY is specified.
- For DELETE_ENTRYLIST requests, the BUFFER area is used as input and should be formatted into 12-byte, 16-byte, or 64-byte elements, where each element is mapped by the IXLYDELI macro and contains all the information required to delete a list entry. The format and size of an element is determined by the options specified on the DELETE_ENTRYLIST request.
  - A 12-byte element is required for any one of the following conditions:
    - VERSCOMPARE=NO and LISTTYPE=IDLIST are specified.
    - VERSCOMPARE=YES and LISTTYPE=IDLIST are specified.
  - A 16-byte element is required for any one of the following conditions:
    - VERSCOMPARE=NO and LISTTYPE=NAMELIST are specified.
    - VERSCOMPARE=YES and LISTTYPE=NAMELIST are specified.
  - A 64-byte element is required when VERSCOMPARE=BYENTRY is specified.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an area (with a length of BUFSIZE) that contains the entry data.

,BUFLIST=*buflist*

Use this output or input parameter to specify a list of buffer addresses to hold data for the request. The set of buffers is used as if it were a single contiguous area.

The format of the list is a set of eight-byte elements. The first four (high-order) bytes of each element are reserved. The second four (low-order) bytes of each element contain the address of a buffer.

There may be from 1 to 16 buffers passed in the list. Each buffer in the list must be the same size and must reside in the same address space or data space. Data is fetched from or stored into the buffers in the order specified.

One of BUFFER or BUFLIST is required for all READ_LIST, READ_MULT, MOVE_ENTRYLIST, and DELETE_ENTRYLIST requests. See the description of the BUFFER keyword for the format of the data contained in the buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 128-byte area that contains the list of buffer addresses.

,BUFNUM=*bufnum*

Use this input parameter to specify the number of buffers in the BUFLIST list. Valid BUFNUM values are from 1 to 16.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the number of buffers in the buffer list.

,BUFSIZE=*bufsize*

Use this input parameter to specify the size of the BUFFER area. See the BUFFER parameter description for valid buffer sizes.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the size of the buffer (BUFFER) in bytes.

,BUFSTGKEY=CALLERS_KEY
,BUFSTGKEY=*bufstgkey*

Use this input parameter to specify a storage key that you define and use when referencing the buffers specified by BUFLIST or the buffer specified by BUFFER.

If you do not specify BUFSTGKEY, or if you specify BUFSTGKEY=CALLERS_KEY, all references to the buffer(s) are performed using the caller’s PSW key.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-bit field that contains the storage key in the format B'kkkkxxxx', where kkkk is the key and xxxx is ignored.

CONTOKEN=*contoken*

Use this input parameter to specify the connect token that was returned by the IXLCONN service. The connect token uniquely identifies your connection to the list structure, and must be specified on each IXLLSTM invocation.

The connect token is available in the IXLCONN answer area mapped by IXLYCONA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the connect token.

,DIRECTION=HEADTOTAIL
,DIRECTION=TAILTOHEAD

Use this input parameter to specify the direction of processing for traversing the given list.

HEADTOTAIL

Processing should begin at the designated entry and proceed toward the tail of the list.

TAILTOHEAD

Processing should begin at the designated entry and proceed toward the head of the list.

,ENTRYID=*entryid*

Use this input parameter to specify the list entry identifier of the entry to be used as the starting point of the READ_LIST or DELETE_LIST request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-byte field that contains the entry identifier.

,ENTRYKEY=*entrykey*

Use this input parameter to either:

- Specify the entry key to be used to compare to the entry key of the list entry to determine if the list entry should be processed. For MOVE_ENTRYLIST and DELETE_ENTRYLIST requests, if the condition specified by KEYREQTYPE is not met for the current list entry, then no processing is performed for the current entry and processing either continues with the next entry to be considered or is terminated based on the value specified for MISCOMPARE. For all other IXLLSTM requests, if the condition specified by KEYREQTYPE is not met for the current list entry, then no processing is performed for the current entry and processing continues with the next entry to be considered.
- For READ_LIST and DELETE_LIST requests, specify the entry key to be used to partially indicate the starting list entry for the request. If DIRECTION=HEADTOTAIL was specified, the designated

starting list entry is the head of the sublist. If DIRECTION=TAILTOHEAD was specified, the designated starting list entry is the tail of the sublist.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry key.

,ENTRYNAME=*entryname*

Use this input parameter to specify the list entry name of the entry to be used as the starting point of the READ_LIST or DELETE_LIST request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field that contains the entry name.

,EXTRESTOKEN=NO_EXTRESTOKEN
,EXTRESTOKEN=*extrestoken*

Use this input parameter to specify a name for an extended restart token specifying an appropriate coupling facility indicator for resuming requests that complete prematurely.

An extended restart token is returned in the LAARESTOKEN answer area specified by ANSAREA when the request terminates prematurely. The extended restart token may be specified on a subsequent READ_MULT or DELETE_MULT request to resume the request at an appropriate point.

The RESTOKEN and EXTRESTOKEN keywords are mutually exclusive. Requestors who specify IXLCONN ALLOWAUTO = YES must use the 16–byte extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16 byte field that contains the extended restart token.

Note: Specifying an extended restart token of all zeros causes the request to consider all entries as unprocessed. Specifying an extended restart token other than one returned from a previous invocation of the request and not fully inititalized to all zeros will produce unpredictable request results.

,FIRSTELEM=1
,FIRSTELEM=*firstelem*

Use this input parameter to specify the index of the first array element to be processed for MOVE_ENTRYLIST and DELETE_ENTRYLIST requests.

The value must reference one of the array elements in the buffers specified by BUFLIST or the buffer specified by BUFFER.

An index value of 1 references the first array element in the BUFFER area or the first BUFLIST buffer.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the index of the first array element to be processed.

,KEYCOMPARE=NO
,KEYCOMPARE=YES

Use this input parameter to specify whether the key value of an existing keyed list entry is to be compared to determine if this entry should be selected for processing.

NO
Specify this option if no entry key comparison will be performed to determine if this entry should be processed.

YES
Specify this option if entry key comparison is to be performed based on the KEYREQTYPE specified to determine if this entry is selectable for processing.

Notes:

1. KEYCOMPARE is only meaningful for list structures allocated on CFLEVEL=1 or higher.
2. KEYCOMPARE=YES is ignored if the target structure does not support keyed entries.

,KEYRANGEEND=*keyrangeend*
Use this input parameter to specify the ending value for the range of keys to be compared to the entry key of the designated list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16 character field that contains the key range ending value.

Note: KEYRANGEEND is a required keyword when KEYREQTYPE=RANGE is specified.

,KEYREQTYPE=EQUAL
,KEYREQTYPE=LESSOREQUAL
,KEYREQTYPE=GREATEROREQUAL
,KEYREQTYPE=RANGE
Use this input parameter to specify how an existing keyed list is located and how entry key comparison is to be performed to determine if the list entry is selectable for processing.

EQUAL
The entry must have a key that equals the ENTRYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the ENTRYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the ENTRYKEY key.

RANGE
The entry must have a key within the specified range of values. The ENTRYKEY specified will be used as the beginning of the range of values. KEYRANGEEND will be used as the ending value. A list entry must have an entry key value within the specified entry key range, inclusive, for it to be selectable.

For a READ_LIST or DELETE_LIST request when LOCATOR=KEYPOS:

1. When no entries on the list meet the requirements of KEYREQTYPE, the request will be failed with a return code X'8' and reason code IXLRSNCODENOENTRY.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, then the entry with an entry key value closest to the value specified will be selected for the starting list entry. When multiple entries have the same entry key value, DIRECTION is used to resolve whether the first or last entry with the entry key value is selected for the starting list entry.
3. When LESSOREQUAL, GREATEROREQUAL or RANGE is specified, if multiple entries have an entry key value within the specified range, DIRECTION will be used to resolve whether the first or last entry within the entry key range is selected for the starting list entry.
4. When KEYREQTYPE=RANGE is specified, KEYRANGEEND=YES is required.

,KEYSCANTYPE=ENTRY
,KEYSCANTYPE=SECONDARY
Use this input parameter to specify which key ordering (entry key ordering or secondary key ordering) will be used to scan for entries on the list.

ENTRY
Entry key ordering will be used for scanning the list.

SECONDARY
Secondary key ordering will be used for scanning the list.

Note: KEYSCANTYPE=SECONDARY is only valid when the structure is allocated in a coupling facility with CFLEVEL=9 or higher.

,KEYTYPE=ENTRY
,KEYTYPE=SECONDARY
Use this input parameter to specify whether to locate the starting list entry using the entry key or the secondary key.

ENTRY
The entry key will be used to locate the starting list entry.

SECONDARY
The secondary key will be used to locate the starting list entry.

Note: KEYTYPE=SECONDARY is only valid when the structure is allocated in a coupling facility with CFLEVEL=9 or higher.

,LASTELEM=*lastelem*
Use this input parameter to specify the index of the last array element to be processed for MOVE_ENTRYLIST or DELETE_ENTRYLIST requests.

The specified value must be greater than or equal to the specified FIRSTELEM value, and must specify one of the array elements passed in the BUFFER area or the BUFLIST buffers.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the index of the last array element to be processed.

,LISTCOMPARE=NO
,LISTCOMPARE=YES
Use this input parameter to specify whether the list number comparison should be used to determine if list entries should be processed.

LISTCOMPARE=NO
List number comparison should not precede processing of list entries.

LISTCOMPARE=YES
List number comparison should precede processing of list entries.

,LISTKEYAREA=NO
,LISTKEYAREA=*listkeyarea*
Use this input parameter to specify the address of a virtual storage area in which entry key values will be places when the current list key value has been assigned to list entries.

List key information will be placed in the LISTKEYAREA when the current list key value has been assigned to the list entry and any of the following conditions exists:

- The request completes successfully.
- The model-dependent timeout has been exceeded.
- A list entry does not exist.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 112-area field that contains an array of entry keys.

,LISTKEYINC=NO
,LISTKEYINC=*listkeyinc*
Use this input parameter to specify a value to be added to the list key after the entry key is set to the list key value.

If the result of adding the value specified by LISTKEYINC to the target list key value is greater than the maximum list key value, the system fails the request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value to be added to the list key after the entry key is set to the list key value.

,LISTNUM=NO_LISTNUM

,LISTNUM=*listnum*

Use this input parameter to specify:

- The number of the list on which the starting list entry resides.
- The number of the list to be compared to the number of the list on which the entries to be processed reside.

If the list comparison fails, then the IXLLSTM operation is terminated and the list entry controls along with the appropriate return and reason codes are provided.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the number of the list.

,LISTTYPE=NAMELIST
,LISTTYPE=IDLIST

Use this input parameter to specify whether the first field in each array element in the BUFFER area or BUFLIST buffers for MOVE_ENTRYLIST or DELETE_ENTRYLIST requests contains an entry name or an EntryID.

LISTYPE=NAMELIST
The first field in each array element in the BUFFER area or BUFLIST buffers contains the entry name of the list entry.

LISTYPE=IDLIST
The first field in each array element in the BUFFER area or BUFLIST buffers contains the EntryID of the list entry. The EntryID may be either system generated or user provided.

Note: LISTTYPE=NAMELIST is not allowed for structures that do not support named entries.

,LOCATOR=CURSOR
,LOCATOR=ENTRYID
,LOCATOR=ENTRYNAME
,LOCATOR=UNKEYPOS
,LOCATOR=KEYPOS

Use this input parameter to specify how to locate the first list entry for READ_LIST or DELETE_LIST requests to be processed.

CURSOR
The list cursor is to be used to designate the starting list entry for the request.

ENTRYID
The EntryID should be used to designate the starting list entry for the request. EntryIDs may be either assigned or provided by the user. User provided EntryIDs must be specified if ENTRYTYPE=USER is specified on the IXLCONN request.

ENTRYNAME
The entry name should be used to designate the starting list entry for the request. ENTRYNAME may only be specified for structures that support named entries.

UNKEYPOS
LISTNUM and DIRECTION will be used to designate the starting entry for the request.

KEYPOS
LISTNUM, DIRECTION and the key specified by KEYTYPE will be used to designate the starting entry for the request.

,LOCKCOMP=NO_LOCKCOMP
,LOCKCOMP=*lockcomp*

Use this input parameter to specify a connection identifier to be verified as the current lock owner as a prerequisite to successful completion of this request.

When LOCKCOMP is specified the locking operation is always considered to be a conditional operation. That is, if the request experiences lock contention the request will be ended with no resultant change to the structure, and appropriate return and reason codes are provided in the ANSAREA. The connection identifier is available from the IXLCONN answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 1-byte field that contains the connection identifier.

,LOCKINDEX=NO_LOCKINDEX
,LOCKCOMP=*lockindex*

Use this input parameter to specify the index of the lock to be operated on within the lock table for the list structure.

When specified, the designated lock will be operated on as specified by the LOCKOPER keyword. The specified value must fall within the range 0 to the number of lock table entries minus one, inclusive.

LOCKINDEX is mutually exclusive with MODE=ASYNCNORESPONSE.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the lock index.

,LOCKOPER=NOTHELD
,LOCKOPER=HELDBY

Use this input parameter to specify the type of operation to be performed on the specified lock.

NOTHELD

The state of the lock must be such that the lock is not held for the duration of the requested list entry operation. The lock state remains unchanged as a result of this option.

When NOTHELD is specified, the locking operation is always considered to be a conditional operation. That is, if the specified lock is held then the entire IXLLSTM operation will be ended with no resultant change to the structure, and appropriate return and reason codes are provided in the ANSAREA.

HELDBY

When LOCKCOMP is not specified, the list operation is to be performed only if the lock is currently held by the connection specified by CONTOKEN.

When LOCKCOMP is specified, the list operation is to be performed only if the lock is currently held by the connection specified by LOCKCOMP.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MISCOMPARE=CONTINUE
,MISCOMPARE=HALT
For MOVE_ENTRYLIST and DELETE_ENTRYLIST requests, use this input parameter to specify whether processing should continue to the next entry or halt when any of the version number, list number or key comparisons are not successful.

CONTINUE
Processing should continue to the next entry if any of the version number, list number or key comparisons, when specified, are not successful.

HALT
If a version number, list number or key comparison is specified but is not successful, processing for the command is stopped. List entry controls and the appropriate return and reason codes will be returned in the ANSAREA.

,MODE=SYNCSUSPEND
,MODE=SYNCECB
,MODE=SYNCEXIT
,MODE=SYNCTOKEN
,MODE=ASYNCECB
,MODE=ASYNCEXIT
,MODE=ASYNCTOKEN
,MODE=ASYNCNORESPONSE
Use this input parameter to specify whether the request is to be performed synchronously or asynchronously.

SYNCSUSPEND
The request will be performed synchronously. Control is not returned to the caller until request processing is complete and the final disposition determined.

If necessary the caller will be suspended until the request completes. To use this option, your program must be enabled for I/O and external interrupts.

SYNCECB
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to the completion of the request, and the ECB specified by REQECB is posted when the request has completed.

When MODE=SYNECB is specified and the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, LISTKEYAREA, ADJAREA, and ANSAREA persist until the REQECB ECB is posted.

SYNCEXIT
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request. When the request completes, the connection's Complete exit will be called.

When the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, ADJAREA, LISTKEYAREA, and ANSAREA persist until the connection's Complete exit is called.

SYNCTOKEN
The request will be attempted synchronously. If the request cannot be completed synchronously, control is returned to the caller prior to completion of the request and a token that uniquely identifies the request is returned.

When the request does not complete synchronously, latent XES binds to the storage locations specified by BUFFER, BUFLIST, ADJAREA, LISTKEYAREA, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

ASYNCECB
The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the ECB specified by REQECB will be posted.

The latent XES binds to the storage locations specified by BUFFER, BUFLIST, LISTKEYAREA, ADJAREA, and ANSAREA persist until the REQECB ECB is posted.

ASYNCEXIT
The request is to be initiated and control is to be returned to the caller prior to completion of the request. When the request completes, the connection's Complete exit will be called.

The latent XES binds to the storage locations specified by BUFFER, BUFLIST, LISTKEYAREA, ADJAREA, and ANSAREA persist until the connection's Complete exit is called.

ASYNCTOKEN
The request is to be initiated, a token generated that uniquely identifies the request on this system, and control returned to the caller prior to completion of the requested operation.

The latent XES binds to the storage locations specified by BUFFER, BUFLIST, LISTKEYAREA, ADJAREA, and ANSAREA persist until a subsequent corresponding IXLFCOMP request indicates completion of the original request.

ASYNCNORESPONSE
The request is to be initiated and control returned to the caller prior to completion of the requested operation. No asynchronous request token is returned; hence no external mechanism exists to force completion of the request.

MODE=ASYNCNORESPONSE is mutually exclusive with LOCKINDEX, BUFFER, and BUFLIST. Any request that does not perform a locking operation and does not use a BUFFER area or BUFLIST buffers may specify MODE=ASYNCNORESPONSE.

,MOVETOKEY=UNCHANGED
,MOVETOKEY=TARGETKEY
,MOVETOKEY=LISTKEY
Use this input parameter to specify on a MOVE_ENTRYLIST request how the key is to be assigned to the list entry when it is moved to the MOVETOLIST. MOVETOKEY may only be specified for structures that support keyed entries.

UNCHANGED
The current entry key value assigned to the list entry will remain unchanged.

TARGETKEY
The TARGET_KEY provided in the array element in the BUFFER or the BUFLIST buffer will be assigned to the list entry when it is moved to the target list.

LISTKEY
The current list key value will be assigned to the list entry when it is moved to the target list.

,MOVETOSKEY=UNCHANGED
,MOVETOSKEY=TARGETKEY
Use this input parameter on MOVE_ENTRYLIST requests to specify how the secondary key is to be assigned to the list entry when it is moved to the MOVELIST.

UNCHANGED
The current secondary key entry value assigned to the list entry will remain unchanged.

TARGETKEY
The TARGET_SKEY provided in the array element in the BUFFER or the BUFLIST buffers will be assigned to the list entry when it is moved to the target list.

Note: MOVETOSKEY can only be specified for structures that support secondary keyed entries.

,PAGEABLE=YES
,PAGEABLE=NO
Use this input parameter to identify whether the storage areas specified by BUFFER or BUFLIST reside in pageable storage.

YES
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in pageable virtual storage.

This includes disabled reference (DREF) storage, and may include storage that has the potential to become pageable during the processing of a request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped during processing of a cache or list request.) It does not include implicitly non-pageable storage (such as is obtained from non-pageable subpools).

The system takes responsibility for managing binds to central storage for the duration of the list request, regardless of what address space owns the storage or whether the storage-owning address space is swappable or nonswappable. The storage can be owned by any address space.

NO
Specify this option to indicate that the BUFFER or BUFLIST buffers reside in non-pageable virtual storage.

This includes implicitly non-pageable storage areas. If the virtual storage may potentially become pageable, the invoker is responsible for ensuring the virtual storage remains non-pageable for the duration of the request. (An example is address space storage owned by any swappable address space, for which a PGSER FIX has been successfully processed, but for which the owning address space gets swapped-out during processing of a list request.)

If MODE=ASYNCTOKEN is specified or MODE=SYNCTOKEN is specified and the request does not complete synchronously, the storage must remain non-pageable until completion of the corresponding IXLFCOMP request. If MODE=ASYNCEXIT is specified or MODE=SYNCEXIT is specified and the request does not complete synchronously, the storage must remain non-pageable until the complete exit is driven for the request. If MODE=ASYNCECB is specified or MODE=SYNCECB is specified and the request does not complete synchronously, the storage must remain non-pageable until the specified ECB is posted for the request.

The system takes responsibility for managing binds to central storage for the duration of the list request, if and only if the non-pageable storage is owned by either the requestor's address space or the connector's address space. If the storage is owned by any other address space, then the invoker is responsible for ensuring that the virtual storage remains non-pageable for the duration of the request (including the case in which the storage is owned by a swappable address space that is swapped during processing of a list request). Subject to this consideration, the storage can be owned by any address space. See *z/OS MVS Programming: Sysplex Services Guide*.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See “Understanding IXLLIST Version Support” on page 668 for a description of the options available with the PLISTVER macro.

,REQDATA=NO_REQDATA
,REQDATA=*reqdata*

Use this input parameter with MODE=SYNCEXIT or MODE=ASYNCEXIT to pass any data you choose to the complete exit. The exit will get control only if the request is processed asynchronously.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the data to be passed to the complete exit.

,REQECB=*reqecb*

Use this output parameter with either MODE=SYNCECB or MODE=ASYNCECB to specify the address of an ECB, which is to be posted when the request completes if the request was processed asynchronously.

Before coding REQECB, you must ensure that:

- You initialize the ECB before you issue the request.
- The ECB resides in either common storage or the home address space where IXLCONN was issued.
- Any tasks that wait for the ECB to be posted reside in the home address space where IXLCONN was issued.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that contains the address of the ECB to be posted when the request completes. The ECB must be aligned on a fullword boundary.

,REQID=NO_REQID
,REQID=*reqid*

Use this input parameter to specify a user-defined request identifier to be associated with the request. You can specify this request identifier on the IXLPURGE macro to cancel a request that has not yet been processed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the user-defined request identifier.

,REQTOKEN=*reqtoken*

Use this output parameter with either MODE=SYNCTOKEN or MODE=ASYNCTOKEN to specify the address of a storage area to receive the request token that is returned when the request will be processed asynchronously. This token, which uniquely identifies the request, must be used as input to the IXLFCOMP macro, which you use to determine if the request has completed.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-byte field where the system will put the request token.

,REQUEST=READ_LIST
,REQUEST=DELETE_LIST
,REQUEST=READ_MULT
,REQUEST=DELETE_MULT
,REQUEST=MOVE_ENTRYLIST
,REQUEST=DELETE_ENTRYLIST

Use this input parameter to specify the type of operation to be performed on the structure.

READ_LIST

Use READ_LIST to request that designated list entries meeting the specified criteria be read into the storage area specified by BUFFER or the buffers specified by BUFLIST. List entries include adjunct data, list entry controls or any combination of these.

DELETE_LIST
Use DELETE_LIST to request that designated list entries meeting the specified criteria be removed from the list on which they reside and returned to the pool of free entries for reuse.

READ_MULT
Use READ_MULT to request that the entry data, the associated adjunct data, or the list entry controls for all allocated entries that meet the criteria specified be read into the storage area specified by BUFFER or the buffers specified by BUFLIST.

DELETE_MULT
Use DELETE_MULT to request that all entries that meet the specified criteria be removed from whatever list they reside on and returned to the pool of free entries for reuse.

MOVE_ENTRYLIST
Use MOVE_ENTRYLIST to request that all specified entries be moved from the current source location to a designated target location. The entries to be moved are found in the list of formatted elements in the storage area specified by BUFFER or the buffers specified by BUFLIST.

DELETE_ENTRYLIST
DELETE_ENTRYLIST is used to request that all specified entries be removed from whichever list they reside on and returned to the pool of free entries for reuse. The entries to be deleted are found in the list of formatted elements in the storage area specified by BUFFER or the buffers specified by BUFLIST.

,RESTOKEN=NO_RESTOKEN
,RESTOKEN=*restoken*
Use this input parameter to specify a name for a restart token specifying an appropriate coupling facility indicator for resuming requests the complete prematurely.

A restart token is returned in the LAARESTOKEN answer area specified by ANSAREA when the request terminates prematurely. The restart token may be specified on a subsequent READ_MULT or DELETE_MULT request to resume the request at an appropriate point.

The RESTOKEN and EXTRESTOKEN keywords are mutually exclusive. Requestors who specify IXLCONN ALLOWAUTO = YES must use the 16–byte extended restart token (EXTRESTOKEN).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8 character field that contains the restart token.

Note: Specifying a restart token of all zeros causes the request to consider all entries as unprocessed. Specifying a restart token other than one returned from a previous invocation of the request and not fully inititalized to all zeros will produce unpredictable request results.

,RETCODE=*retcode*
Use this output parameter to specify a field to contain the return code. (The return code is also returned in register 15.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the return code when the request has completed.

,RSNCODE=*rsncode*
Use this output parameter to specify a field to contain the reason code returned, if applicable. (The reason code is also returned in register 0.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4-byte field that will contain the reason code (if any) when the request has completed.

,SECONDARYKEY=*xsecondarykey*
Use this input parameter to either:

- Specify the secondary key value to be compared to the secondary key of the list entry to determine if the list entry should be processed. For MOVE_ENTRYLIST and DELETE_ENTRYLIST requests, if the condition specified by SKEYREQTYPE is not met for the current list entry, then no processing is performed for the current entry and processing either continues with the next entry to be considered or is terminated based on the value specified for MISCOMPARE. For all other request types, if the condition specified by SKEYREQTYPE is not met for the current list entry, then no processing is performed for the current entry and processing continues with the next entry to be considered.
- Specify the secondary key to be used to partially indicate the starting list entry for the request for READ_LIST and DELETE_LIST requests. If DIRECTION=HEADTOTAIL was specified, the designated starting list entry is the head of the sublist. If DIRECTION=TAILTOHEAD was specified, the designated starting list enty is the tail of the sublist.

For a READ_LIST or DELETE_LIST request when LOCATOR=KEYPOS:

1. When no entries on the list meet the requirements of SKEYREQTYPE, the request will be failed with a return code X'8' and reason code IXLRSNCODENOENTRY.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have an entry key value equal to the specified value, but entries exist with an entry key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, then the entry with an entry key value closest to the value specified will be selected for the starting list entry. When multiple entries have the same entry key value, DIRECTION is used to resolve whether the first or last entry with the entry key value is selected for the starting list entry.
3. When LESSOREQUAL, GREATEROREQUAL or RANGE is specified, if multiple entries have an entry key value within the specified range, DIRECTION will be used to resolve whether the first or last entry within the entry key range is selected for the starting list entry.
4. When KEYREQTYPE=RANGE is specified, KEYRANGEEND=YES is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32 character field that contains the secondary key.

Note: SECONDARYKEY is required if SKEYCOMPARE=YES.

,SKEYCOMPARE=NO
,SKEYCOMPARE=YES

Use this input parameter to specify whether the secondary key value of an existing keyed list entry should be compared to the SKEYREQTYPE to determine if this entry should be selected for processing.

NO

Specify this value if no secondary key comparison will be performed to determine if this entry should be processed.

YES

Specify this option if secondary key comparison is to be performed based on the SKEYREQTYPE to determine if this entry is to be selected for processing.

Note: SKEYCOMPARE=YES is ignored if the target structure was not allocated with secondary keys.

,SKEYRANGEEND=*skeyrangeend*

Use this input parameter to specify the ending value for the range of keys to be compared to the secondary key of the designated list entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 32 character field that contains the secondary key range ending value.

Note: SKEYRANGEEND must be specified when SKEYREQTYPE=RANGE.

,SKEYREQTYPE=EQUAL

,SKEYREQTYPE=LESSOREQUAL
,SKEYREQTYPE=GREATEROREQUAL
,SKEYREQTYPE=RANGE

Use this input parameter to specify how an existing keyed list entry is located and how key comparison is to be performed to determine if the list entry is selectable for processing.

EQUAL
The entry must have a key that equals the SECONDARYKEY key.

LESSOREQUAL
The entry must have a key that is less than or equal to the SECONDARYKEY key.

GREATEROREQUAL
The entry must have a key that is greater than or equal to the SECONDARYKEY key.

RANGE
The entry must have a key within the specified range of values. The SECONDARYKEY specified will be used as the beginning of the range of values. SKEYRANGEEND will be used as the ending value. A list entry must have an secondary key value within the specified entry key range, inclusive, for it to be selectable.

Notes:

1. When no entries on the list meet the requirements of the SKEYREQTYPE, and a new entry cannot be created, the request will be failed, and the appropriate return and reason codes are provided to the invoker.
2. When LESSOREQUAL or GREATEROREQUAL are specified, if no entries on the list have a secondary key value equal to the specified value, but entries exist with a secondary key value greater than (if GREATEROREQUAL was specified), or less than (if LESSOREQUAL was specified) the entry key value specified, then the entry with a secondary key value closest to the value specified will be selected. When multiple entries have the same secondary key value, LISTPOS is used to resolve whether the first or last entry with the secondary key value is selected.
3. When LESSOREQUAL, GREATEROREQUAL or RANGE is specified, if multiple entries have a secondary key value within the specified range, DIRECTION will be used to resolve whether the first or last entry within the secondary key range is selected.
4. When SKEYREQTYPE=RANGE is specified, KEYRANGEEND=YES is required.

,TYPE=ENTDATA
,TYPE=ADJDATA
,TYPE=ECONTROLS

Use this group of required input(s) to specify the type of information to be read. Any combination of ENTDATA, ADJDATA and ECONTROLS may be specified.

ENTDATA
Use ENTDATA to indicate that entry data is to be read.

ADJDATA
Use ADJDATA to indicate that adjunct data is to be read.

ECONTROLS
Use ECONTROLS to indicate that list entry control information is to be read.

Notes:

1. ADJDATA is only functional for structures that support adjunct data.
2. For structures that are allocated with secondary keys, the first 32 bytes of the adjunct data will contain the secondary key information.
3. For structures allocated with secondary keys, the secondary key is not read by TYPE=ECONTROLS. The secondary key is read by TYPE=ADJDATA.

,VERSCOMP=NO_VERSCOMP

,VERSCOMP=*verscomp*

Use this input parameter to specify a version number to be compared to the version number of the existing entry. If this request creates a new entry, the VERSCOMP specification is ignored. The existing entry is processed only if its version number meets the condition specified by the VERSCOMPTYPE parameter. If the condition specified by VERSCOMPTYPE is not met for the current list entry, then no processing is performed for the current list entry and processing continues with the next entry to be considered.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-byte field that contains the version number.

,VERSCOMPARE=NO
,VERSCOMPARE=YES

Use this input parameter to specify if the version number is to be compared to the version number of the existing entry.

VERSCOMPARE=NO
No version comparison should be performed to determine if each list entry should be processed.

VERSCOMPARE=YES
Version number comparison should precede processing of each list entry.

,VERSCOMPTYPE=EQUAL
,VERSCOMPTYPE=LESSOREQUAL

Use this input parameter to specify how a list entry version number comparison as specified by VERSCOMP is to be performed.

Note: The VERSCOMPTYPE parameter is valid only for list structures allocated in a coupling facility with CFLEVEL=1 or higher.

VERSCOMPTYPE=EQUAL
The version number for the list entry must be equal to the value specified for VERSCOMP.

VERSCOMPTYPE=LESSOREQUAL
The version number for the list entry must be less than or equal to the value specified for VERSCOMP.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and RETCODE, if specified) contains a return code.
- If the return code is not zero, GPR 0 (and RSNCODE, if specified) contains a reason code.

Note: The return and reason codes will also be put into the answer area (mapped by IXLYLAA), if there is one, when the request completes.

The IXLYCON macro provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table contains hexadecimal return and reason codes, the equate symbols associated with each reason code, and the meaning and suggested action for each return and reason code.

Table 76. Return and Reason Codes for IXLLSTM Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** If you specified a MODE value of ASYNCECB, ASYNCEXIT, ASYNCTOKEN, or ASYNCNORESPONSE, the request has been successfully initiated. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. For any other MODE value, the request has successfully completed. **Action:** • If you specified MODE=ASYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=ASYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=ASYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.
4	xxxx0402	**Equate Symbol:** IXLRSNCODEASYNCH **Meaning:** The request could not be processed synchronously. It will be processed asynchronously. The answer area (ANSAREA) fields will not contain valid information until the asynchronous processing has completed. • If MODE=SYNCECB was specified, the ECB specified for REQECB will be posted when the request has finished. • If MODE=SYNCEXIT was specified, the connection's complete exit will be given control when the request has finished. • If MODE=SYNCTOKEN was specified, an asynchronous request token was returned in the area specified by REQTOKEN. This request token may be specified on the IXLFCOMP macro to determine when the request has finished. **Action:** • If you specified MODE=SYNCECB, wait on the ECB you specified in REQECB. • If you specified MODE=SYNCEXIT, the connection's complete exit will be given control when the request has completed. • If you specified MODE=SYNCTOKEN, use REQTOKEN when calling IXLFCOMP to determine when the request has completed.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** The request completed prematurely because it exceeded the coupling facility model-dependent time-out criteria. The following information has been returned in the answer area: • The number of output elements returned in the BUFFER area or BUFLIST buffers (for READ_LIST and READ_MULT). • The number of deleted structure entries (for DELETE_MULT, DELETE_LIST and DELETE_ENTRYLIST). • A restart token or an extended restart token (for READ_MULT and DELETE_MULT). • The list entry controls for the first unprocessed entry in the list sequence (for READ_LIST and DELETE_LIST). • The index of the first unprocessed entry indentifier or name (for DELETE_ENTRYLIST and MOVE_ENTRYLIST). **Action:** After processing all returned data, reissue the request to continue. For more information about premature completion, of an IXLLSTM request, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx040D	**Equate Symbol:** IXLRSNCODEBADREADADJDATA **Meaning:** Program error. The request specified that adjunct data was to be read, but the storage area specified by ADJAREA is not addressable. All other requested data was retrieved and the following fields in the answer area have been filled in: • LAARLRMLCTLS - The first processed entry in the list sequence. • LAAREADCNT - The number of entries processed successfully • LAALCTL (if the request completed prematurely as well) - The list entry controls for the first unprocessed entry in the list sequence. **Note:** Only the adjunct data for the first entry in the list is placed in the ADJAREA. The buffers should contain the adjunct data for the other entries in the list. **Action:** • Verify the ADJAREA address. • ADJAREA must be addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTM while disabled, ADJAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTM in AR-mode and you specified the ADJAREA address using explicit register notation, the corresponding access register must be updated appropriately. • If you are calling IXLLSTM in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTM macro. • If LAALCTL is not zeros, the request completed prematurely. See the action suggested for return code X'4' reason code X'xxxx0409'. Correct the address specified by ADJAREA, and rerun the request asking for adjunct data only.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** Program error. The request has completed prematurely because the buffer specified by BUFFER or the buffers specified by BUFLIST are full. For both IXLLSTM READ_LIST and READ_MULT requests, the number of output elements returned in the BUFFER area or BUFLIST buffers has been returned in the answer area. For READ_LIST, the list entry controls for the first unprocessed entry in the list sequence has been returned in the answer area. **Action:** Reissue the request, or increase the size of the buffer(s), and rerun your program. You can reissue the request using the restart token (RESTOKEN) or extended restart token (EXTRESTOKEN). If the same buffers are used, the data returned from the original request will be overwritten so be sure to process the returned data first. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
4	xxxx0410	**Equate Symbol:** IXLRSNCODELOCKCOND **Meaning:** For a LOCKOPER=HELDBY request, or a request that specified LOCKCOMP, the request could not be completed successfully because the specified lock is not currently held as required. The connection identifier of the lock owner is returned in the answer area (LAACONID field). **Action:** Retry the request, or obtain the lock as required, and retry the request. If you are unable to get the lock, check the ID in the LAACONID field and determine if some recovery is necessary.
4	xxxx0412	**Equate Symbol:** IXLRSNCODELOCKHELDBYSYS **Meaning:** The lock is not held by any connection, but instead is held by the system. For an LOCKOPER=HELDBY request, the request could not be completed successfully because the specified lock is not currently held as required: **Action:** Retry the request, or obtain the lock as required, and retry the request.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Program error. The specified connect token (CONTOKEN) was not valid for one of the following reasons: 1. The user with the connection identifier represented by the token has disconnected from the structure. 2. The connector's task (the task that issued IXLCONN) ended. 3. The specified token is not the token that was returned from IXLCONN. 4. The request was issued from an address space other than the address space in which IXLCONN was issued. 5. The connect token was invalidated during rebuild. 6. The connect token was invalidated by XES. **Note:** The answer area (ANSAREA) fields are not valid. **Action:** Take the action with the corresponding meaning. 1. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know of the disconnection. 2. Discontinue use of the structure. Perform recovery and cleanup for the structure and check the protocol for the use of the structure to determine why you did not know that the task ended. 3. The contoken is returned in the CONACONTOKEN field of the answer area specified on the IXLCONN request. 4. Issue your request from the same address space the IXLCONN was issued in. 5. Wait for the rebuild to complete, and try again. 6. Discontinue use of the structure. Perform recovery and cleanup for the structure.
8	xxxx0822	**Equate Symbol:** IXLRSNCODEBADREADTYPE **Meaning:** Program error. You specified that either entry data or adjunct data should be returned, but the list structure does not contain the type of data you requested. No data is returned. **Action:** Ensure that you are requesting the type of data you need from the structure and that the structure supports that type of data. Issue IXLMG to get more information about the structure.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	**Equate Symbol:** IXLRSNCODEWRONGSTRTYPE **Meaning:** Program error. The connection specified by CONTOKEN is not to a list structure. **Action:** Specify the appropriate CONTOKEN value, and rerun your program. The CONTOKEN is returned in the answer area provided on the IXLCONN invocation. The type of structure built is passed on the TYPE parameter of the IXLCONN macro.
8	xxxx0825	**Equate Symbol:** IXLRSNCODENOENTRY **Meaning:** Program error. The designated list entry does not exist; therefore, no entries were processed. **Action:** None necessary. However, if this return code and reason code are unexpected, you should check the way the list entry was created. Examine the parameters specified for locating the list entry on the invocation of this macro.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. The value specified for either FIRSTELEM or LASTELEM was not valid. Some entries may have been processed. When FIRSTELEM or LASTELEM is not valid, the index of the first entry identifier or name that was not processed is returned in the answer area. **Action:** Ensure that the FIRSTELEM and LASTELEM values correspond to entries in the list of entries to be processed.
8	xxxx0833	**Equate Symbol:** IXLRSNCODEBADPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list is specified as being pageable (PAGEABLE=YES), but is not. **Action:** Change the buffer area(s) to pageable storage, or specify PAGEABLE=NO. See the PAGEABLE parameter description for specification instructions.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0834	**Equate Symbol:** IXLRSNCODEBADNONPGBLATTR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is specified as being nonpageable (PAGEABLE=NO), but is either pageable or not addressable. **Action:** Ensure that: • The type of storage in the buffer area corresponds to the value you specified on the PAGEABLE parameter. See the PAGEABLE parameter description for specification instructions. • The correct buffer address was used. • The buffer area was not previously freed. • If you are calling IXLLSTM while disabled, the buffers must reside in either page-fixed or DREF storage. • The buffer areas were allocated in a storage key that matches the key specified by the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR-mode: – If the BUFFER or BUFLIST parameter was specified using explicit register notation, the corresponding access register was updated appropriately. – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the IXLLSTM macro.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by BUFFER, or one of the buffers in the BUFLIST list, is not addressable. **Action:** Ensure that: • The correct buffer address was used. • The buffer area was not previously freed. • The buffer area was allocated in a storage key that matches the key specified using the BUFSTGKEY parameter, or, if BUFSTGKEY is omitted, the caller's PSW key. • If BUFLIST was specified and your program is running in AR mode: – The BUFALET specification is correct. – You specified SYSSTATE ASCENV=AR before issuing the macro.
8	xxxx0836	**Equate Symbol:** IXLRSNCODEBADREALADDR **Meaning:** Program error. Real storage addresses were provided in the BUFLIST list, but one of the buffers is not addressable in central storage. **Action:** • Verify that BUFADDRTYPE was specified as you intended. • Ensure that the buffer addresses specified by BUFLIST are valid.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol:** IXLRSNCODEBADANSAREA **Meaning:** Program error. The storage area specified by ANSAREA is not addressable. **Action:** Ensure that: • The answer area address specified by ANSAREA is valid. • If the caller is running in AR-mode and the ANSAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for ANSAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTM while disabled, ANSAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTM in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTM macro.
8	xxxx0839	**Equate Symbol:** IXLRSNCODEBADREQTOKENAREA **Meaning:** Program error. The storage area specified by REQTOKEN is not addressable. **Action:** Ensure that: • The request token area specified by REQTOKEN is valid. • If the caller is running in AR-mode and the REQTOKEN parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLLSTM while disabled, REQTOKEN must reside in either page-fixed or DREF storage. • If you are calling IXLLSTM in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTM macro.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient to contain answer area information. The answer area (ANSAREA) fields do not contain valid information. **Action:** Increase the size of the answer area, and rerun your program. The minimum size of the ANSAREA is in the LAA_LEN field of the IXLYLAA mapping macro.
8	xxxx083E	**Equate Symbol:** IXLRSNCODEMAXLISTKEY **Meaning:** Program error. The list key to be assigned to an entry that was being created or moved was not valid. Either the list key or the list key plus the list key increment value was greater than the maximum list key. **Action:** Ensure that you specified a correct list key increment. Depending on the protocol you are using for assigning list key values, you might want to issue a WRITE_LCONTROLS request to update either the list key value to a lower value or the maximum list key value to a higher value.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx083F	**Equate Symbol:** IXLRSNCODEBADENTRYVERSION **Meaning:** The entry designated by the specified list entry controls has a version number that does not meet the criteria specified by VERSCOMP and VERSCOMPTYPE. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if this return code and reason code are unexpected, you might want to verify the values specified in VERSCOMP and VERSCOMPTYPE and look at the version returned in the answer area.
8	xxxx0840	**Equate Symbol:** IXLRSNCODEBADENTRYLIST **Meaning:** The entry designated by the specified list entry controls does not reside on the list specified by LISTNUM. The list entry controls for the entry are returned in the answer area (field LAALCTL). **Action:** None necessary. However, if you did not expect this return and reason code, you might want to verify what list this entry is on. Maybe the entry was moved, or you designated the wrong list in LISTNUM. Check the protocol for using this list. The information returned in the LAALCTL field of the answer area contains the number of the list that this list entry is on as well as other control information.
8	xxxx0841	**Equate Symbol:** IXLRSNCODEBADENTRYNAME **Meaning:** Program error. The name specified by ENTRYNAME is not a unique name within the structure, and therefore entry creation is suppressed. **Action:** Be sure to provide a unique entry name when creating entries to be written to structures that support entry names.
8	xxxx0842	**Equate Symbol:** IXLRSNCODEPERSISTENTLOCK **Meaning:** Program error. The request specifying a NOTHELD lock operation failed because the lock was held by a connection that is in the failed-persistent state. The connection identifier of the lock owner is returned in the answer area (field LAACONID). **Action:** Either perform recovery for the connection, or wait until recovery for the connection is performed.
8	xxxx0843	**Equate Symbol:** IXLRSNCODEBADENTRYID **Meaning:** Program error. The entry corresponding to either an entry identifier or entry name in either IDLIST or NAMELIST, respectively, does not exist. The index to the failing entry identifier or name was returned in the answer area (field LAAFAILINDEX). The count of entries deleted is also returned in the answer are (field LAADELCNT). **Action:** Remove the failing entry from the list, or reissue the request. Update FIRSTELEM to point after the failing entry (FIRSTELEM = LAAFAILINDEX + 1) to the next entry on the list to be processed.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	**Equate Symbol:** IXLRSNCODENONAMES **Meaning:** Program error. A list entry was designated by entry name, but the structure does not support entry names. **Action:** Ensure that you are connected to the intended structure. Ensure that the correct specification was made for REFOPTION on the IXLCONN macro. You may want to issue IXLMG to get more information about the structure.
8	xxxx0846	**Equate Symbol:** IXLRSNCODEBADLOCKINDEX **Meaning:** Program error. The specified LOCKINDEX exceeds the size of the lock table for the structure. **Action:** Correct LOCKINDEX to specify an index that is contained within the lock table. The maximum value for the LOCKINDEX is one less than the value specified by the LOCKENTRIES parameter on the IXLCONN macro.
8	xxxx0847	**Equate Symbol:** IXLRSNCODEBADLISTNUMBER **Meaning:** Program error. The specified LISTNUM value exceeds the number of lists for the structure. **Action:** Correct LISTNUM to specify a list number that exists in the structure. The number of lists allocated for a structure is determined on the LISTHEADERS parameter of the IXLCONN macro. The list numbers go from 0 to n-1, where n is the number of lists specified.
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. The restart token specified by RESTOKEN is not valid. **Action:** Determine why the restart token cannot be used to restart the request. Possible causes are: • The specified token does not correspond to the restart token returned in the answer area of the previous request (field LAARESTOKEN). • The user specified RESTOKEN when EXTRESTOKEN was required. • The user specified EXTRESTOKEN when RESTOKEN was required. For more information about premature request completion, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx084A	**Equate Symbol:** IXLRSNCODENOKEYS **Meaning:** Program error. The structure does not support the use of entry keys. The IXLLSTM request type either required the structure to support entry keys or designated a sublist, list entry, or list position by list number and entry key. **Action:** Ensure that you are connected to the intended structure. The use of keys for a structure is determined by the REFOPTION keyword on the IXLCONN macro.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx084B	**Equate Symbol:** IXLRSNCODENOLOCKS **Meaning:** Program error. A locking operation was requested, but the structure does not contain a lock table. **Action:** Ensure that: • You are connected to the intended structure. • You intended to perform a locking operation. The use of locks is determined by the LOCKENTRIES keyword on the IXLCONN macro.
8	xxxx084E	**Equate Symbol:** IXLRSNCODEBADMOVETOLIST **Meaning:** Program error. The list number specified for MOVETOLIST exceeds the number of lists defined for the structure. **Action:** Correct MOVETOLIST to specify a list that exists in the structure. The maximum number of lists in a structure is determined by the LISTHEADERS parameter on the IXLCONN macro.
8	xxxx0851	**Equate Symbol:** IXLRSNCODENOSUSPENDISABLE **Meaning:** Program error. The request failed because MODE=SYNCSUSPEND was specified, but the caller is disabled and cannot be suspended. **Action:** Either specify another MODE value, or become enabled (release the CPU lock), then reissue the request.
8	xxxx0854	**Equate Symbol:** IXLRSNCODEBADLOCKCOMP **Meaning:** Program error. The connection identifier specified for LOCKCOMP is not valid. **Action:** The connection identifier is returned in the answer area (mapped by IXLYCONA) when the IXLCONN macro was issued.
8	xxxx0859	**Equate Symbol:** IXLRSNCODEBADLISTAUTH **Meaning:** The list authority value for the specified list does not meet the criteria specified by AUTHCOMP and AUTHCOMPTYPE. The current list authority (field LAALISTAUTH) and description (field LAALISTDESC) are returned in the answer area. **Action:** None required; however you might want to take some action depending on your application. The list authority is set to binary zeros when the structure is allocated. When IXLLSTM is issued with the NEWAUTH keyword, the list authority is changed if the correct comparison list authority value is specified. Check the answer area to determine the list authority value for the list specified. Verify that the correct list number was specified.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0864	**Equate Symbol:** IXLRSNCODEBADBUFSIZE **Meaning:** Program error. The size of the BUFFER area or the buffer areas specified by BUFLIST is not large enough to contain the data being read. No data is returned. **Action:** If more space is available, specify a larger buffer size, and reissue the request. For READ_LIST and READ_MULT requests, the answer area contains the list entry controls for the first list entry selected for processing.
8	xxxx0865	**Equate Symbol:** IXLRSNCODEBADBUFSPEC **Meaning:** Program error. There is an error in the buffer specification. **Action:** Check the following: • If BUFLIST was specified, check the requirements for BUFLIST, BUFNUM, and BUFINCRNUM. • If BUFFER was specified, check the requirements for BUFFER and BUFSIZE. • Buffer pointer(s) in BUFLIST • Buffer boundaries.
8	xxxx0866	**Equate Symbol:** IXLRSNCODEBADBUFKEY **Meaning:** Program error. The buffer storage key specified by BUFSTGKEY is incorrect. For requests that write coupling facility data, the data cannot be fetched from the specified buffer area. For requests that read coupling facility data, the data cannot be stored into the specified buffer area. **Action:** Check the following: • Determine if the key of the storage being used for the buffers is different from the PSW key. • If BUFSTGKEY was specified, verify that the key was put in the correct bits (see the explanation of this parameter for more information). • If you are calling IXLLSTM in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLLSTM macro.
8	xxxx0867	**Equate Symbol:** IXLRSNCODEBADBUFLIST **Meaning:** Program error. The 128-byte storage area specified by BUFLIST is not addressable. **Action:** Ensure that the address specified by BUFLIST is valid.
8	xxxx086A	**Equate Symbol:** IXLRSNCODEBADELEMNUM **Meaning:** Program error. The value specified for ELEMNUM is not valid. **Action:** Correct the ELEMNUM specification to be within the allowable range: from zero to whatever MAXELEMNUM value was returned to the connector in the connect answer area.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. The extended restart token specified by EXTRESTOKEN is not valid. The specified token refers to an older instance of the target structure. **Action:** Reinitiate the request with an EXTRESTOKEN value of zero. A system-managed process occurred between the time a request returned the extended restart token and the time the connector tried to continue the request using that token. Discard the results of the initial request and reissue the request. For more information about restarting requests, see *z/OS MVS Programming: Sysplex Services Guide*.
8	xxxx0894	**Equate Symbol:** IXLRSNCODEBADKEYCOMPARE **Meaning:** Program error. The value specified for ENTRYKEY used with the KEYCOMPARE=YES with KEYREQTYPE did not match. The value specified for SECONDARYKEY used with SKEYCOMPARE=YES with SKEYREQTYPE did not match. **Action:** Ensure that the values specified for either ENTRYKEY or SECONDARYKEY are valid.
8	xxxx0895	**Equate Symbol:** IXLRSNCODEBADKEYLISTAREA **Meaning:** Program error. The storage area specified for LISTKEYAREA is not addressable. **Action:** Ensure that: • The list key area address specified by LISTKEYAREA is valid. • If the caller is running in AR-mode and the LISTKEYAREA parameter was specified using explicit register notation, the corresponding access register was updated appropriately. • The address specified for LISTKEYAREA is addressable in the caller's primary address space or from the caller's PASN access list. • If you are calling IXLLSTM while disabled, LISTKEYAREA must reside in either page-fixed or DREF storage. • If you are calling IXLLSTM in AR-mode, SUSSTATE ASCENV=AR must be issued before the IXLLSTM macro.
8	xxxx0897	**Equate Symbol:** IXLRSNCODEBADKEYTYPE **Meaning:** Program error. The specified KEYTYPE value is not valid for the specified structure. **Action:** Ensure that the value of KEYTYPE is valid.
8	xxxx0898	**Equate Symbol:** IXLRSNCODEBADKEYSCANTYPE **Meaning:** Program error. The specified KEYSCANTYPE value is not valid for the specified structure. **Action:** Correct the value of KEYSCANTYPE to reflect the use of either entry keys or secondary keys.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0899	**Equate Symbol:** IXLRSNCODEBADSKEYCOMPARE **Meaning:** Program error. The specified SKEYCOMPARE value is not valid for the specified structure. **Action:** Correct the value of SKEYCOMPARE.
8	xxxx089A	**Equate Symbol:** IXLRSNCODEBADSKEYREQTYPE **Meaning:** Program error. The specified SKEYREQTYPE value is not valid for the specified structure. **Action:** Correct the value of SKEYREQTYPE.
8	xxxx089B	**Equate Symbol:** IXLRSNCODEBADKEYCOMPARETYPE **Meaning:** Program error. The specified KEYCOMPARE value is not valid for the specified structure. **Action:** Correct the value of KEYCOMPARE.
8	xxxx089C	**Equate Symbol:** IXLRSNCODEBADMOVETOKEY **Meaning:** Program error. The specified MOVETOKEY value is not valid for the specifed structure. **Action:** Ensure that the value of MOVETOKEY is correct.
8	xxxx089D	**Equate Symbol:** IXLRSNCODEBADMOVETOSKEY **Meaning:** Program error. The specified MOVETOSKEY value is not valid for the specifed structure. **Action:** Ensure that the value of MOVETOSKEY is correct.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. This system does not have connectivity to the coupling facility that contains the list structure. Possible reasons for this are: • The operator issued VARY PATH,OFFLINE. • The operator issued CONFIG CHP,OFFLINE. • Hardware errors to the coupling facility. • Facility or path failure to the coupling facility. **Action:** Begin rebuilding the structure on a different coupling facility, or disconnect from the structure.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C14	**Equate Symbol:** IXLRSNCODESTATUSUNKNOWN **Meaning:** Environmental error. The request has completed, but the final disposition of the request cannot be determined. The answer area (ANSAREA) fields do not contain valid information. **Action:** • Verify the validity of your data by comparing the expected results with what is in the coupling facility. • Consider running rebuild processing to move the structure to attempt to protect it from further errors of this kind.
C	xxxx0C18	**Equate Symbol:** IXLRSNCODELISTFULL **Meaning:** Environmental error. The entry could not be placed on the designated target list because the list is full. **Action:** Either change the maximum number of entries allowed on the list by specifying a new LISTLIMIT on a WRITE_LCONTROLS request. You should be monitoring the usage of the list structure every time a read or write is done. (LAATOTALCNT is the total count of allocated entries in the list structure, and LAATOTALELECNT is the total count of allocated elements in the list structure.) This data should be evaluated periodically to ensure structure resources are being used efficiently.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The list structure failed prior to completion of the request. **Action:** Either rebuild or disconnect from the structure.
C	xxxx0C68	**Equate Symbol:** IXLRSNCODEBADREQCFLEVEL **Meaning:** Environmental error. The request type is not permitted for the level of coupling facility in which the target structure is allocated. **Action:** Either disconnect from the structure (using IXLDISC) or initiate a rebuild of the structure, if allowed (using IXLREBLD) in a coupling facility of the correct CFLEVEL.
C	xxxx0CA0	**Equate Symbol:** IXLRSNCODEQUIESCEDSUSPENDFAIL **Meaning:** Environmental error. The request is failed because the structure is quiesced for a system-managed process and SUSPEND=FAIL is specified on the IXLCONN. **Action:** None, if this is expected.
C	xxxxFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This could occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.

Table 76. Return and Reason Codes for IXLLSTM Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
10	xxxx10xx	**Meaning:** System error. XES processing failure. The state of the involved structure and the disposition of the request are unpredictable. The answer area (ANSAREA) fields do not contain valid information. **Action:** Contact the IBM support center.

IXLMG — Coupling Facility Measurement

Description

The IXLMG macro allows an authorized caller to request measurement data related to the use of a coupling facility. The information is available by coupling facility or by structure. Information on subchannels and connectivity is returned when any coupling facility information is requested.

Structures in the IXLYAMDA mapping macro provide the format for the data:
- IXLYAMDHD maps common area for header entries
- IXLYAMDAREA maps the header record
- IXLYAMDCF and IXLYAMDCF1 map coupling facility information
- IXLYAMDSTRL and IXLYAMDSTRL1 map coupling facility list structure information
- IXLYAMDSTRC and IXLYAMDSTRC1 map coupling facility cache structure information
- IXLYAMDSC and IXLYAMDSC1 map coupling facility subchannel information
- IXLYAMDSLL and IXLYANDSLL1 map structure limits for a list structure
- IXLYAMDSLC and IXLYAMDSLC1 map structure limits for a cache structure
- IXLYAMDCFMI maps coupling facility measurement header information
- IXLYAMDCFMINFO maps coupling facility measurement information
- IXLYAMDCFRF maps coupling facility remote facility information
- IXLYAMDSCSC and IXLYANDSCSC1 map structure information for cache storage classes
- IXLYAMDSCOC maps structure information for cache cast-out classes
- IXLYAMDSCOCSTATS maps structure information for cache cast-out classes
- IXLYAMDSSCC maps the structure copy controls record.

The data returned may be system-oriented data (system attachment information or statistics on the system's use of a structure) or sysplex-wide data (data that is retrieved from the coupling facility itself).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB for CFNAME invocations Task for STRNAME invocations
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	No locks held or only the CPU lock held
Control parameters:	Must be in the primary address space

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before invoking IXLMG. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

The IXLYAMDA macro provides the format of the area that DATAAREA points to. Include that macro in your program.

Restrictions

The input parameter list must be addressable in the caller's primary address space.

Input Register Information

Before issuing the IXLMG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLMG macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if applicable, if GPR 15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLMG macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro, and restore them after the system returns control.

Performance Implications

IXLMG processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLMG Version Support

The IXLMG macro supports versions in the range of 0 - 2.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLMG macro.
- The following keyword is supported by all versions starting with version 1 and higher of the IXLMG macro.

AMDALEVEL

- The following keyword is supported by all versions starting with version 2 and higher of the IXLMG macro.

STRCOPYCNTLS

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLMG macro is as follows:

main diagram

```
►►──IXLMG──b──DATAAREA=dataarea──,DATALEN=datalen──────────────────────────────────►

                                          ┌─,STRCOPYCNTLS=NO──┐
                     ┌─,HWSTATISTICS=YES──┤                   ├─┐
   ┌─,CFNAME=ALL_CFS─┐                    └─,STRCOPYCNTLS=YES─┘ │
►──┼─────────────────┼──┼─────────────────────────────────────────┼──┼──┬────────────────────┬──►
   │ └─,CFNAME=cfname──┘  ├─,HWSTATISTICS=NO──────────────────────┤  │  ├─,AMDALEVEL=0─────────┤
   │                      └─,HWSTATISTICS=CF──────────────────────┘  │  └─,AMDALEVEL=amdalevel─┘
   └─┤ parameters-1 ├──────────────────────────────────────────────┘

                                    ┌─,PLISTVER=IMPLIED_VERSION─┐
►─,RETCODE=retcode──,RSNCODE=rsncode─┼───────────────────────────┼──────────────────►
                                    ├─,PLISTVER=MAX─────────────┤
                                    └─,PLISTVER=plistver────────┘

   ┌─,MF=S──────────────────────────────────┐
   │                   ┌─,0D──────┐         │
►──┼─,MF=(L,mfctrl─────┴─,mfattr──┴─)───────┼──────────────────────────────────────►◄
   │                   ┌─,COMPLETE─┐        │
   └─,MF=(E,mfctrl─────┼───────────┼─)──────┘
                       └─,COMPLETE─┘
```

parameters-1

```
                         ┌─,COCLASSB=NO_COCLASSB─────────────────────┐ ┌─,STGCLASS=NO_STGCLASS─┐
►►──,STRNAME=strname─────┼───────────────────────────────────────────┼─┼───────────────────────┼──────►◄
                         └─,COCLASSB=coclassb,COCLASSE=coclasse──────┘ └─,STGCLASS=stgclass────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,AMDALEVEL=<u>0</u>
,AMDALEVEL=*amdalevel*

Use this input parameter to specify the level of the IXLYAMDA record mappings that the system returns in the answer area. Valid values are 0 and 1.

- A value of 0 indicates that base level IXLYAMDA records will be returned.
- A value of 1 indicates that level-1 IXLYAMDA records will be returned. To request a level-1 IXLYAMDA record, use version 1 of the IXLMG macro by specifying AMDALEVEL=1.

,CFNAME=<u>ALL_CFS</u>
,CFNAME=*cfname*

Use this input parameter to specify the name of the coupling facility for which data is to be gathered or to indicate that data is to be collected for all coupling facilities that are attached to the system on which the IXLMG macro is issued. If not specified, information about all coupling facilities that have a configured attachment to the system on which the IXLMG macro is invoked will be gathered. If CFNAME specifies a coupling facility name, the coupling facility and all allocated structures will be included in the reported data.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the eight-character field that contains the name of the coupling facility for which data is to be gathered.

,COCLASSB=NO_COCLASSB
,COCLASSB=*coclassb*
Use this input parameter to specify the first cast-out class to be reported on for a cache structure. The specified value must fall within the range of 1 to the maximum coupling facility class value defined for the structure, inclusive. (The number of cast-out classes is determined on the IXLCONN macro.)

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the two-byte field that contains the first cast-out class in the range of cast-out classes to be reported on.

,COCLASSE=*coclasse*
Use this input parameter to specify the last cast-out class to be reported on for a cache structure. The specified value must fall within the range 1 to the maximum cast-out class value defined for the structure, inclusive, and be greater than or equal to the value for COCLASSB.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a two-byte field that contains the last cast-out class in the range of cast-out classes to be reported on.

DATAAREA=*dataarea*
Use this output parameter to identify the name of the output storage area that is to contain the measurement data returned by the measurement gathering routine. The storage area is defined by the macro IXLYAMDA.

The storage area provided must be in fixed or disabled reference storage.

This macro does not clear the caller's storage area. It is the caller's responsibility to clear the area, if this is required.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the name of the storage area to contain the measurement data.

,DATALEN=*datalen*
Use the input parameter to specify the length of the storage area to contain the measurement data returned by the measurement gathering routine.

Use the IXLYAMDA macro to determine the length of the DATAAREA by determining the length of each section that you request and how many times each section will be repeated.

The minimum length is the length of the IXLYAMDA header. If the DATAAREA is not large enough to hold all the data requested, a return code X'4' with a reason code of IXLRSNCODEMOREDATA will be returned. The length of the output data will be returned in the IXLYAMDA, in the field IXLYAMDAREA_TLEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the length of the storage area.

,HWSTATISTICS=YES
,HWSTATISTICS=NO
,HWSTATISTICS=CF
Use this input parameter to indicate whether the statistics gathered from the coupling facility should be included in the measurement data. The statistics include coupling facility measurement information and structure control information.

YES
Include statistics gathered from the coupling facility in the returned measurement data.

NO
Do not include statistics gathered from the coupling facility in the returned measurement data.

CF
Include statistics gathered from the coupling facility only for coupling facility measurement information, and not for structure control information.

Notes:

1. If HWSTATISTICS=YES is specified, the data will include information retrieved from the coupling facility, specifically structure control information for each structure and measurement data for each coupling facility. The data returned will contain information on all allocated structures in the coupling facility whether the structure has an active connector from the system on which IXLMG was invoked or not. This information could contain structures not currently known in the CFRM active policy.
   - The data returned for structures with active connectors from the system on which IXLMG was invoked contains both structure usage and control information.
   - The data returned for structures with no active connectors from the system on which IXLMG was invoked contains only structure control information.
   - The data returned for each coupling facility contains coupling facility usage, control, and measurement information.
2. If HWSTATISTICS=NO is specified, the data will not include structure control information or facility measurement information retrieved from the coupling facility.
   - The data returned for structures with active connectors from the system on which IXLMG was invoked contains structure usage information.
   - No data is returned for structures with no active connectors from the system on which IXLMG was invoked.
   - The data returned for each coupling facility contains coupling facility usage and control information.
3. If HWSTATISTICS=CF is specified, the data will include measurement data for each coupling facility retrieved from the coupling facility but will not include the structure control information for each structure.
   - The data returned for structures with active connectors from the system on which IXLMG was invoked contains structure usage information.
   - No data is returned for structures with no active connectors from the system on which IXLMG was invoked.
   - The data returned for each coupling facility contains coupling facility usage, control, and measurement information.

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl*,*mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLMG Version Support” on page 1158 for a description of the options available with PLISTVER.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,STGCLASS= NO_STGCLASS
,STGCLASS=*stgclass*
Use this input parameter to specify for a cache structure the storage class for which you would like storage class statistics.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte field that contains the storage class for which storage class statistics should be reported.

,STRCOPYCNTLS=NO
,STRCOPYCNTLS=YES
Use this input parameter to specify whether the structure copy controls record should be included in the measurement data associated with the structure records.

Note that this information is intended for system use only, and therefore no mapping is provided for the structure copy controls data.

NO
Structure copy controls should not be included in the measurement data.

YES
Structure copy controls should be included in the measurement data.

Specifying STRCOPYCNTLS=YES requires an AMDALEVEL of 1 or higher.

,STRNAME=*strname*
Use this input parameter to specify the name of the coupling facility structure for which data is to be gathered.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the name of the coupling facility structure for which data is to be gathered.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the system returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code.

Table 77. Return and Reason Codes for the IXLMG Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None	**Meaning:** IXLMG completed successfully and returned the requested information in the data area provided. **Action:** None.
4	xxxx0404	**Equate Symbol:** IXLRSNCODEMOREDATA **Meaning:** Not all requested data could be returned in the data area provided because the data area was not large enough. **Action:** Retry the request with a larger data area. The required size is provided in the IXLYAMDA header in the field IXLYAMDAREA_TLEN.
4	xxxx0421	**Equate Symbol:** IXLRSNCODESTGCLASSERR **Meaning:** The requested data could not be returned for the storage class information. The STGCLASS specified is a storage class that exceeds the maximum defined storage class for the structure. The maximum defined storage class is defined on the IXLCONN for initial structure allocation. **Action:** Retry the request with the correct value.
4	xxxx0422	**Equate Symbol:** IXLRSNCODECOCLASSERR **Meaning:** Not all requested data could be returned for the cast-out class information. Either the COCLASSE was larger than the maximum cast-out class, or the COCLASSB was larger than the maximum cast-out class for the structure. The start and end of range values returned indicates the amount of data reported. The cast-out class values are set on the IXLCONN invocation. **Action:** Retry the request with correct start and end values.
4	xxxx0423	**Equate Symbol:** IXLRSNCODESTRUCTUREERR **Meaning:** No data could be returned for the structure specified. The name is not for a known structure. **Action:** Retry the request with the correct structure name.

Table 77. Return and Reason Codes for the IXLMG Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0426	**Equate Symbol:** IXLRSNCODESTRUCTUREFAIL **Meaning:** No data could be returned for the structure specified. The structure is in the failed state. **Action:** None necessary. IXLMG should only be called for valid structures. If this is unexpected, check your protocol to determine why you did not know about the problem with this structure.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** The ALET for the parameter list is not valid. **Action:** All parameters must be in the primary address space so all ALETs should be set up accordingly.
8	xxxx0803	**Equate Symbol:** IXLRSNCODERESERVEDNOT0 **Meaning:** Reserved field in parameter list is not zero. **Action:** Check to see if your program overlaid the parameter list. Check the parameter list address to make sure it is uncorrupted. Make sure you are running with the same version of MVS that the macro was compiled with.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

Table 77. Return and Reason Codes for the IXLMG Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080D	**Equate Symbol:** IXLRSNCODEAREATOOSMALL **Meaning:** DATAAREA is too small to contain the minimum response information. (The minimum response information is the IXLYAMDA header information mapped by IXLYAMDAREA.) **Action:** Run the request again and specify a DATAAREA with a minimum DATALEN of IXLYAMDAREA_LENGTH.
8	xxxx080E	**Equate Symbol:** IXLRSNCODEBADAREA **Meaning:** Error accessing DATAAREA. **Action:** Check that the DATAAREA specified is addressable in the primary address space. Make sure the DATAAREA is in fixed or disabled reference storage.
8	xxxx080F	**Equate Symbol:** IXLRSNCODEBADAREAALET **Meaning:** DATAAREA ALET is not valid. **Action:** All parameters must be in the primary address space so all ALETs should be set up accordingly.
8	xxxx0885	**Equate Symbol:** IXLRSNCODEBADAMDALEVEL **Meaning:** The value specified for AMDALEVEL is not valid. **Action:** Retry the request with the correct AMDALEVEL value.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environment error. There are no coupling facility services available. The hardware support necessary to provide coupling facility services might not be present. **Action:** XES may not be used when XCF is running in local mode. XES requires that the coupling facility be installed.
10	xxxx10xx	**Meaning:** Software failure in XES software processing. **Action:** Contact the IBM support center.

IXLPURGE — Purge Operations to a Coupling Facility

Description

The IXLPURGE macro allows an authorized caller to complete operations in progress to a coupling facility and purge operations not yet processed. The operations include requests initiated by the IXLCACHE, IXLLIST, and IXLRT services.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PKM allowing key 0 -
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or Access Register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space

Programming Requirements

If the program is in AR mode, issue SYSSTATE ASCENV=AR before invoking IXLPURGE. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

None.

Input Register Information

Before issuing the IXLPURGE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLPURGE macro, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code, if applicable, if GPR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLPURGE macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control. For asynchronous processing of requests, see *z/OS MVS Programming: Sysplex Services Guide*.

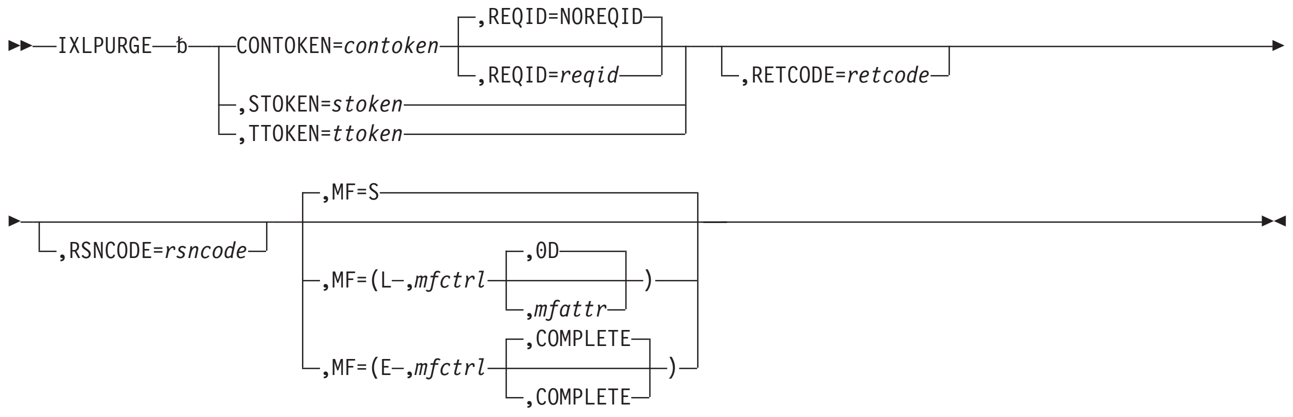
Performance Implications

None.

Syntax Diagram

The syntax of the IXLPURGE macro is as follows:

main diagram



Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

CONTOKEN=*contoken*

Use this input parameter to specify the connect token that identifies the connector this request is to be processed for. All requests for that connection will be purged. (The connect token is in the IXLYCONA answer area, after the IXLCONN is issued, in the CONACONTOKEN field.)

To Code: Specify the name or address (using a register from 2 - 12) of the 16-character storage area that contains the connector token.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,REQID=NOREQID
,REQID=*reqid*
Use this input parameter with the CONTOKEN keyword to identify the request or requests that are associated with the connection to be processed. This parameter corresponds to the REQID parameter specified on an IXLLIST or IXLCACHE request. Only those requests that were initiated with this REQID and the indicated CONTOKEN will be purged.

To Code: Specify the name or address (using a register from 2 - 12) of the 8-character storage area that contains the request identifier.

STOKEN=*stoken*
Use this input parameter to specify the name of the token that identifies the address space for which the request is to process. All requests associated with the address space will be purged.

To Code: Specify the name or address (using a register from 2 - 12) of the 8-character storage area that contains the address space token.

TTOKEN=*ttoken*
Use this input parameter to specify the name of the token that identifies the task for which the request is to process. All requests associated with the task will be purged.

To Code: Specify the name or address (using a register from 2 - 12) of the 16-character storage area that contains the task token.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 78. Return and Reason Codes for the IXLPURGE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None	**Meaning:** Purge is successful. **Action:** None.

Table 78. Return and Reason Codes for the IXLPURGE Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list for this request is not addressable. The answer area (ANSAREA) fields are not valid. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are issuing this macro while disabled, the parameter list resides in either page-fixed or DREF storage. • If you are issuing this macro in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are issuing this macro in AR-mode, SYSSTATE ASCENV=AR must be issued before issuing this macro.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSIONNUM (or IXLRSNCODEBADVERSION#) **Meaning:** The version number in the macro parameter list is not compatible with the level of XES currently being used. The answer area (ANSAREA) fields are not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.

IXLREBLD — Rebuilding or Duplexing a Structure

Description

Use the IXLREBLD macro to control the rebuild process for a coupling facility structure. There are two types of rebuild processing, rebuild and duplexing rebuild. The method by which the rebuild processing is accomplished can be either user-managed or system-managed.

- User-managed rebuild allows you to allocate another structure with the same name in the same or a different coupling facility for the purpose of planned reconfiguration and recovery. In a structure rebuild, the user is responsible for reconstructing data in the newly allocated structure. Rebuild allows you to change the location or attributes of a structure.
- User-managed duplexing rebuild allows you to allocate another cache structure in a different coupling facility for the purpose of duplexing the data in the structure to achieve better availability and usability. In a user-managed duplexing rebuild, the user is responsible for managing both instances of the structure and when necessary, can revert to using only one of the structures. User-managed duplexing rebuild is valid only for cache structures.
- System-managed rebuild allows the system to provide the support necessary for rebuild processing. Connectors recognize that the structure is temporarily unavailable for requests and are not active participants in the rebuild process. System-managed rebuild requires that active connectors have specified IXLCONN ALLOWAUTO=YES. It is also recommended that ALLOWALTER=YES be specified when connecting to the structure.

  System-managed rebuild provides the capability for persistent structures with no connectors or with only failed-persistent connectors to be rebuilt.

  System-managed rebuild is valid for all types of structures; however, if a user-managed rebuild implementation is already in place, the system will give precedence to that method of rebuild.
- System-managed duplexing rebuild allows the system to allocate another structure in a different coupling facility for the purpose of duplexing the data in the structure. System-managed duplexing rebuild provides a recovery mechanism by which rapid failover to the unaffected structure instance of a duplexed pair can occur in the event of a failure. System-managed duplexing rebuild does not require connectors to be participants in the duplexing process, although the connectors recognize that the structure is temporarily unavailable for requests, but does require that connectors be aware of its progress. System-managed duplexing rebuild is valid for all types of structures; however, if a user-managed duplexing rebuild implementation is already in place, the system will give precedence to that method of duplexing.

See *z/OS MVS Programming: Sysplex Services Guide* for a complete description of rebuild processing and its requirements.

Phases of the user-managed rebuild process are reported to the event exits of the users coordinating the structure rebuild. IXLREBLD requestors are not required to be connected to a structure to start or stop a structure rebuild. However, only an active connector to the structure can issue the IXLREBLD request to indicate that the rebuild process is complete.

Phases of the system-managed rebuild process are managed by the system on behalf of the connector. System-managed rebuild does not require that there be any active connectors to the structure.

You can use IXLREBLD to do the following processes:
- Start rebuilding.
- Stop rebuilding.
- Indicate that you have completed rebuild processing. (User-managed rebuild only)
- Start duplexing rebuild
- Stop duplexing rebuild
- Indicate that you have completed duplexing rebuild processing. (User-managed duplexing rebuild only)

For starting and stopping a rebuild process, you must provide a reason for the operation. The reason is passed to the event exits of the users participating in the rebuild process.

You initiate the rebuild process by issuing an `IXLREBLD REQUEST=START...` request for either an individual structure or for all structures in a particular coupling facility.

You initiate the duplexing rebuild process by issuing an `IXLREBLD REQUEST=STARTDUPLEX...` request for either an individual structure or for all structures in a particular coupling facility. Depending on the method of duplexing, if any, supported by each structure, the system will start either user-managed or system-managed duplexing rebuild appropriate for each structure. User-managed duplexing rebuild supports only cache structures; system-managed duplexing rebuild supports all structure types.

Another use for the IXLREBLD macro is to populate a coupling facility with a set of appropriate structures after the coupling facility has been removed from the configuration for service and then returned to the sysplex configuration. The POPULATECF function, available with OS/390 Release 6 and higher, provides a means of redistributing multiple structures in a coupling facility with a single IXLREBLD macro invocation or SETXCF command. The advantage of the POPULATECF function is that it rebuilds structures that are currently allocated in a coupling facility into the coupling facility that the structure has specified (in the CFRM policy) is preferable. If the structure already resides in a coupling facility that is higher in its preference list than the coupling facility being populated, the system will not rebuild the structure. See *z/OS MVS Setting Up a Sysplex* for a description of how to use the POPULATE function.

The security administrator may have controlled the use of installation structures through the use of RACF or another security product. If so, ensure that you are authorized to issue the IXLREBLD macro for the structure.

For more information about the structure rebuild process, see *z/OS MVS Programming: Sysplex Services Guide*.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN, any SASN. For IXLREBLD REQUEST=COMPLETE and IXLREBLD REQUEST=DUPLEXCOMPLETE, the primary address space must be equal to the requestor's primary address space at the time of the connection.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLREBLD. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Restrictions

None.

Input Register Information

Before issuing the IXLREBLD macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code, if applicable, if GPR 15 return code is nonzero
1	Used as a work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLREBLD macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Performance Implications

The rebuild process entails coordination among active connections to achieve the reconstruction of the structure. The performance of IXLREBLD will depend on the size of the structure, the number of connections to the structure, and the amount of data to be reconstructed. Also, depending on the protocols of connections during IXLREBLD, the services provided by those connections may be temporarily unavailable.

IXLREBLD processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLREBLD Version Support

The IXLREBLD macro supports versions 0, 1 and 2.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLREBLD macro.
- The following keyword is supported by all versions starting with version 1 and higher of the IXLREBLD macro.

LESSCONNACTION

- The following keyword is supported by all versions starting with version 2 and higher of the IXLREBLD macro.

POPULATECF

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLREBLD macro is as follows:

main diagram

```
►►──IXLREBLD──b──REQUEST=──┬─START──┤ parameters-1 ├──────────────┬──┬──────────────────┬──────────────►
                           ├─STOP──┤ parameters-2 ├───────────────┤  └─,RETCODE=retcode─┘
                           ├─COMPLETE───,CONTOKEN=contoken────────┤
                           ├─STARTDUPLEX─┤ parameters-3 ├─────────┤
                           ├─STOPDUPLEX─┤ parameters-4 ├──────────┤
                           └─DUPLEXCOMPLETE───,CONTOKEN=contoken──┘

                           ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────┐
►──┬──────────────────┬────┼───────────────────────────┼──┼────────────────────────────────────┼──►◄
   └─,RSNCODE=rsncode─┘    ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D──────┐      │
                           └─,PLISTVER=plistver────────┘  ├─,MF=(L─,mfctrl──┼──────────┼─)────┤
                                                          │                  └─,mfattr──┘      │
                                                          │                  ┌─,COMPLETE─┐     │
                                                          └─,MF=(E─,mfctrl──┼───────────┼─)───┘
                                                                             └─,COMPLETE─┘
```

parameters-1

```
                                                  ┌─,LOCATION=NORMAL─┐  ┌─,LESSCONNACTION=TERMINATE─┐
►►───┬─,STRNAME=strname──────────┬────────────────┼──────────────────┼──┼───────────────────────────┼──────►
     ├─,CFNAME=cfname────────────┤                └─,LOCATION=OTHER──┘  └─LESSCONNACTION=CONTINUE───┘
     └─,POPULATECF=populatecf────┘

►─,STARTREASON=──┬─LOSSCONN──────────────────────────────────────┬──────────────────────────────────►◄
                 ├─STRFAILURE────────────────────────────────────┤
                 ├─CONNECTOR──┬───────────────────────┬──────────┤
                 │            └─,USERCODE=usercode────┘          │
                 └─OPERATOR───,CART=cart────,CONSID=consid───────┘
```

parameters-2

```
►►───┬─,STRNAME=strname──────────┬──,STOPREASON=──┬─LOSSCONNOLD─────────────────────────────────┬──────►◄
     ├─,CFNAME=cfname────────────┤                ├─LOSSCONNNEW─────────────────────────────────┤
     └─,POPULATECF=populatecf────┘                ├─STRFAILUREOLD───────────────────────────────┤
                                                  ├─CONNECTOR──┬──────────────────────┬─────────┤
                                                  │            └─,USERCODE=usercode───┘         │
                                                  └─OPERATOR───,CART=cart────,CONSID=consid─────┘
```

parameters-3

```
►►───┬─,STRNAME=strname──┬──,STARTREASON=──┬─CONNECTOR──┬──────────────────────┬─────────┬──────────►◄
     └─,CFNAME=cfname────┘                 │            └─,USERCODE=usercode───┘         │
                                           └─OPERATOR───,CART=cart────,CONSID=consid─────┘
```

parameters-4

```
                                                    ┌─,IGNOREDUPLEX=NO──┐
►►──┬─,STRNAME=strname──┬─,KEEP=NEW───────┬─┬───┼───────────────────┼───────────────────────────────►
    │                   └─,──KEEP──=──OLD─┘ │   └─,IGNOREDUPLEX=YES─┘
    └─,CFNAME=cfname────────────────────────┘

►─,STOPREASON=─┬─LOSSCONN──────────────────────────────────┬─────────────────────────────────────►◄
               ├─STRFAILURE────────────────────────────────┤
               ├─CONNECTOR─┬───────────────────────┬───────┤
               │           └─,USERCODE=usercode─┘        │
               └─OPERATOR──,CART=cart────,CONSID=consid──┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,CART=*cart*
: Use this input parameter to specify command-and-response token (CART) associated with the console to be notified when any rebuild- or duplexing rebuild-related start or completion occurs. The system reports a Rebuild Quiesce event with OPERATOR as the reason and places the CART value in the event exit parameter list IXLYEEPL (field EEPLCART) for each connected user.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the eight-character input field that contains the command-and-response token (CART).

,CFNAME=*cfname*
: Use this input parameter to specify the name of a coupling facility in which structure rebuild is requested for every structure allocated other than XCF signalling structures. Note that this function is intended only for use by a cleanup utility. The system will not start rebuild for XCF signalling structures in the named coupling facility. The user should rebuild XCF signalling structures in a coupling facility one at a time.

  The coupling facility name must be 8 characters long, padded on the right with blanks if necessary. The name may contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_), and must begin with an uppercase alphabetic character.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 8-character input field that contains the coupling facility name.

,CONSID=*consid*
: Use this input parameter to specify the system-generated console ID for the console or extended MCS console to be notified when any rebuild- or duplexing rebuild-related start or completion occurs.

  The system reports a Rebuild Quiesce event with OPERATOR as the reason, and places the console id of the issuing console in the event exit parameter list IXLYEEPL (field EEPLCONSID) for each connected user.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the console ID.

,CONTOKEN=*contoken*
: Use this input parameter to specify the original connect token returned to the requestor by the IXLCONN macro when connecting to the structure.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the connect token.

,IGNOREDUPLEX=<u>NO</u>
,IGNOREDUPLEX=YES

Use this input parameter to specify whether to ignore the DUPLEX specification in the CFRM active policy to restart a duplexing rebuild at the completion ot the stop processing. A specification of DUPLEX(ENABLED) in the CFRM active policy means that MVS will attempt to automatically restart a duplexing rebuild when the duplexing rebuild completes or is stopped.

IGNOREDUPLEX=NO
Do not ignore the DUPLEX specification in the CFRM active policy. MVS should automatically restart duplexing rebuild if appropriate.

IGNOREDUPLEX=YES
Ignore the DUPLEX specification in the CFRM active policy. MVS should not automatically restart duplexing rebuild even if the CFRM active policy specifies DUPLEX(ENABLED). This does not prevent a duplexing rebuild from being started at a later time after the stop completion processing. The DUPLEX(ENABLED) specification in the CFRM active policy should be changed to either DUPLEX(DISABLED) or DUPLEX(ALLOWED) if you do not want MVS to automatically initiate duplexing rebuild.

Note that this override of the CFRM policy DUPLEX specification only applies to the completion of this duplexing rebuild stop processing. If DUPLEX(ENABLED) is in effect, MVS may later initiate duplexing rebuild automatically.

,KEEP=NEW
,KEEP=OLD
Use this input parameter to specify which structure should remain after the duplexing rebuild has been stopped.

KEEP=NEW
Duplexing rebuild should stop and switch to using the new structure.

KEEP=OLD
Duplexing rebuild should stop to fall back to using the old structure.

,LESSCONNACTION=<u>TERMINATE</u>
,LESSCONNACTION=CONTINUE
Use this input parameter to indicate whether XES is to allow a rebuild to continue, in spite of a degradation in connectivity to the new structure.

When a structure rebuild is requested, XES evaluates whether the new structure will have the same or better connectivity than the current structure, as determined by the Sysplex Failure Management (SFM) weights of the connector's systems attached to the coupling facility. If the evaluation shows that the new structure will have poorer connectivity than the current structure, XES uses the LESSCONNACTION specification to determine what action to take in regard to the structure rebuild.

Notes:

1. If the IXLREBLD invocation specifies REASON=LOSSCONN, XES ignores the LESSCONNACTION keyword and processes the request as if LESSCONNACTION=TERMINATE were specified.
2. For duplexing rebuild requests, LESSCONNACTION=TERMINATE is assumed; the system will not initiate a duplexing rebuild if the new structure has poorer connectivity than the old structure, as determined by the SFM weights of the systems connected to the structure in the coupling facility.
3. XES ignores the LESSCONNACTION keyword when a rebuild start request results in system-managed processing. If any active connector loses connectivity to the structure being rebuilt, the system-managed process always stops.

LESSCONNACTION=TERMINATE
The system is to stop the rebuild if the new structure has poorer connectivity than the old structure, as determined by the SFM weights of the systems connected to the structure in the coupling facility.

LESSCONNACTION=CONTINUE
The system is not to stop the structure rebuild. Users that cannot connect to the new structure must disconnect from the structure to allow the rebuild to continue, or must stop the rebuild.

Notes:

1. If LESSCONNACTION is not specified when the rebuild is started, the default action is to not to continue any rebuild, despite its effect on the connectors to the structure.
2. If LESSCONNACTION=CONTINUE is specified with the CFNAME= keyword, all structures in the specified coupling facility are rebuilt, despite any degradation in connectivity for the structures being rebuilt.

,LOCATION=NORMAL
,LOCATION=OTHER
Use this input parameter to indicate where the new structure that is allocated for rebuild can be located.

NORMAL
The new structure can be allocated in any coupling facility in the preference list using the normal allocation rules.

OTHER
The new structure cannot be allocated in the same coupling facility as the original structure. The new structure can be allocated in any coupling facility in the preference list, other than the coupling facility containing the original structure, using normal allocation rules. LOCATION=OTHER is assumed when the rebuild is a duplexing rebuild.

This option is intended for planned reconfiguration cases where a system administrator has verified that the policy does contain a coupling facility where the structure can be rebuilt. If there is only one coupling facility in the preference list for the structure being rebuilt and the coupling facility is the location of the original structure, each connector will receive reason code IXLRSNCODENOFAC when the rebuild connect is attempted. In IXLYCONA, CONAFACILITYARRAY will indicate that the current coupling facility was not chosen because the LOCATION=OTHER option was specified on IXLREBLD.

Note: A structure rebuild initiated with STARTREASON=LOSSCONN will always be stopped if the connectivity will not be improved for the current set of active connectors by the rebuild process. LOCATION=OTHER is NOT implied by a LOSSCONN rebuild, as there may not be another viable coupling facility available in which to rebuild the structure.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See “Understanding IXLREBLD Version Support” on page 1173 for a description of the options available with PLISTVER.

,POPULATECF=*populatecf*
Use this input parameter to specify the name of the coupling facility which is to be populated with structures selected from those currently allocated in other less preferred coupling facilities.

A structure rebuild will be attempted for each allocated structure in the policy that contains the specified coupling facility in its preference list, if the specified coupling facility is at a higher position in the preference list than the coupling facility in which the structure currently is allocated. If the structure already is allocated in a more preferable coupling facility, the rebuild is not started.

Each selected structure will be placed in a ‘pending rebuild’ state; these ‘pending rebuild’ structures are then processed serially to completion (either a stop complete event or a rebuild complete event is received) before the system selects the next structure for rebuild.

This option is intended for use by an operations product or subsystem only. It is not intended for use by an application initiating rebuilds for structures owned by other applications.

The coupling facility name must be 8 characters long, padded on the right with blanks if necessary. The name may contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_), and must begin with an uppercase alphabetic character.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 8-character field that contains the name of the coupling facility.

REQUEST=START
REQUEST=STOP
REQUEST=COMPLETE
REQUEST=STARTDUPLEX
REQUEST=STOPDUPLEX
REQUEST=DUPLEXCOMPLETE
Use this input parameter to specify the type of rebuild request.

- Use REQUEST=START to request start rebuild processing for the specified structure or structures within the specified coupling facility. The request will start non-duplexing rebuild processing, which might be either user-managed or system-managed.

- Use REQUEST=STOP to request stop rebuild processing for the specified structure or structures within the specified coupling facility. The request will stop non-duplexing rebuild processing only, either for a user-managed or system-managed rebuild.
- Use REQUEST=COMPLETE to indicate that user-managed rebuild processing is complete for the specified connection.
- Use REQUEST=STARTDUPLEX to request start duplexing rebuild processing for the specified structure or structures within the specified coupling facility.
- Use REQUEST=STOPDUPLEX to request stop duplexing rebuild processing for the specified structure or structures within the specified coupling facility. The request implies that the connector is reverting to simplex mode by either switching to the new structure or falling back to the old structure.
- Use REQUEST=DUPLEXCOMPLETE to indicate that duplexing rebuild processing is complete for the specified connection.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword output field to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword output field to contain the reason code.

,STARTREASON=LOSSCONN
,STARTREASON=STRFAILURE
,STARTREASON=CONNECTOR
,STARTREASON=OPERATOR
Use this input keyword to specify the reason for starting the rebuild or duplexing rebuild process for the structure.

STARTREASON=LOSSCONN
Loss of connectivity to the structure (for rebuild processing only)

STARTREASON=STRFAILURE
Structure failure (for rebuild processing only)

STARTREASON=CONNECTOR
Connected user-specified reason. The requestor can specify the USERCODE parameter to indicate the reason for initiating the rebuild.

STARTREASON=OPERATOR
Operator-initiated rebuild processing. This option requires that you also provide a command-and-response token (CART) and console ID (CONSID) associated with the console to be notified of the rebuild progress.

,STOPREASON=LOSSCONNOLD
,STOPREASON=LOSSCONNNEW
,STOPREASON=LOSSCONN
,STOPREASON=STRFAILUREOLD
,STOPREASON=STRFAILURE
,STOPREASON=CONNECTOR
,STOPREASON=OPERATOR,CART=*cart***,CONSID=***consid*
Use this input parameter to specify the reason for stopping the rebuild or the duplexing rebuild process for the structure. You can specify one of the following reasons:

STOPREASON=LOSSCONNOLD
Loss of connectivity to the original structure (for rebuild processing only)

STOPREASON=LOSSCONNNEW
Loss of connectivity to the new structure (for rebuild processing only)

STOPREASON=LOSSCONN
Loss of connectivity to the structure (for duplexing rebuild processing only)

STOPREASON=STRFAILUREOLD
Failure of the original structure (for rebuild processing only

STOPREASON=STRFAILURE
Failure of the structure (for duplexing rebuild processing only)

STOPREASON=CONNECTOR
Connected user-specified reason. The requestor can specify the USERCODE parameter to indicate the reason for stopping the rebuild.

STOPREASON=OPERATOR
Operator-initiated request to stop rebuild processing. This option requires that you also provide a command-and-response token (CART) and console ID (CONSID) associated with the console to be notified of the rebuild processing.

,STRNAME=*strname*
Use this input parameter to specify the name of the structure. The structure name must be 16 characters long, padded on the right with blanks if necessary. The name may contain numeric characters, uppercase alphabetic characters, national characters ($, @, #), or an underscore (_).

Users can use the IXCQUERY service macro to determine what structures have been defined in the active CFRM policy.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the name of the target structure.

,USERCODE=0
,USERCODE=*usercode*
Use this input parameter to specify the user code that represents the connector's reason for initiating rebuild processing.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the user code representing the connector's reason for rebuilding.

Return and Reason Codes

When the IXLREBLD macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 79. Return and Reason Codes for the IXLREBLD Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** IXLREBLD request was successful. **Action:** None.
4	xxxx0415	**Equate Symbol:** IXLRSNCODEALREADYREBUILDING **Meaning:** Structure rebuild or duplexing rebuild for the structure has already been initiated. The REBUILD REQUEST=START REQUEST=STARTDUPLEX request is not processed. **Action:** Examine your protocol for initiating the rebuild process.
4	xxxx0416	**Equate Symbol:** IXLRSNCODEALREADYSTOPPING **Meaning:** Structure rebuild or duplexing rebuild stop has already been initiated. The REBUILD REQUEST=STOP or REQUEST=STOPDUPLEX request is not processed. **Action:** Examine your protocol for stopping the rebuild process.
4	xxxx041D	**Equate Symbol:** IXLRSNCODEIGNOREFORREBUILDSTOP **Meaning:** The system ignores IXLREBLD REQUEST=COMPLETE or REQUEST=DUPLEXCOMPLETE because a rebuild stop or stopduplex request for the structure is in progress. The request fails. **Action:** Examine your rebuild protocol.
4	xxxx0425	**Equate Symbol:** IXLRSNCODENOSTRFOUND **Meaning:** The system did not find any structures eligible for rebuild in the specified coupling facility. The request fails. **Action:** Verify that the names of the structure and the coupling facility have been specified correctly.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The IXLREBLD parameter list is not accessible. **Action:** Verify that: • The parameter list address is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are calling IXLREBLD in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register was updated appropriately. • If you are calling IXLREBLD in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLREBLD.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** The IXLREBLD parameter list ALET is not accessible. **Action:** Ensure that the ALET is zero or that the ALET represents a valid entry on the DU-AL.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** There is an invalid version number in the IXLREBLD parameter list. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS that your program is running on.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** The requestor is in SRB mode. **Action:** Do not issue the IXLREBLD macro when running in SRB mode.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** The requestor is not in an enabled state. **Action:** Issue the IXLREBLD macro while running enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** The requestor's primary address space is not the same as the home address space. **Action:** Make sure that the primary address space and the home address space are the same at the time of the IXLREBLD invocation.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** The requestor specified a CONTOKEN that is not valid. The contoken is invalid for one of the following reasons: disconnect has occurred, EOT of the connector's task, input contoken is not the contoken returned from IXLCONN, or the request was issued outside the connector's address space. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure.
8	xxxx084C	**Equate Symbol:** IXLRSNCODENOSAFAUTH **Meaning:** The requestor does not have the required SAF authorization. **Action:** Determine why the installation has not allowed the proper SAF authorization for access to the structure.
8	xxxx0863	**Equate Symbol:** IXLRSNCODETASKTERM **Meaning:** The request is rejected because the requesting task is going through termination. IXLREBLD cannot be issued from a resource manager. **Action:** Examine your protocol to ensure that IXLREBLD is not issued from a resource manager.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0886	**Equate Symbol:** IXLRSNCODEBADREQUEST **Meaning:** The IXLREBLD REQUEST type is not supported. **Action:** Ensure that the request type has been specified correctly and that you are running on an appropriate level of z/OS.
C	xxxx0C05	**Equate Symbol:** IXLRSNCODESTRNOTINPOLICY **Meaning:** Environmental error. The structure specified is not defined in the active CFRM policy. **Action:** Ensure that the structure name has been specified correctly.
C	xxxx0C07	**Equate Symbol:** IXLRSNCODECFNOTINPOLICY **Meaning:** Environmental error. The coupling facility specified is not defined in the active CFRM policy. **Action:** Ensure that the coupling facility name has been specified correctly.
C	xxxx0C0A	**Equate Symbol:** IXLRSNCODESTRNOTALLOCATED **Meaning:** Environmental error. The structure specified is not allocated. **Action:** Ensure that the structure name has been specified correctly.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. The rebuild request has been rejected because the structure has failed. A rebuild request that would result in a system-managed rebuild will be rejected if the structure has failed. **Action:** Examine the protocol in use by the application for recovery from structure failure.
C	xxxx0C29	**Equate Symbol:** IXLRSNCODEXESNOTACTIVE **Meaning:** Environmental error. CFRM services are not active or are not available. **Action:** Bring the CFRM couple data set in use in the sysplex.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C35	**Equate Symbol:** IXLRSNCODENOACTIVECONNS **Meaning:** Environmental error. There are no active connections to the structure. The request to rebuild the structure is rejected. **Action:** Ensure that there is at least one active connection to the structure that is to be rebuilt.
C	xxxx0C36	**Equate Symbol:** IXLRSNCODESTOPINPROGRESS **Meaning:** Environmental error. The rebuild start or startduplex request for the structure fails because processing to stop rebuilding the same structure name is in progress. **Action:** Examine your rebuild protocol.
C	xxxx0C3D	**Equate Symbol:** IXLRSNCODENOTREBUILDING **Meaning:** Environmental error. The request to rebuild the structure is rejected because the structure is not in the rebuild process. **Action:** Ensure that the structure name was specified correctly.
C	xxxx0C3E	**Equate Symbol:** IXLRSNCODEINCLEANUP **Meaning:** Environmental error. The rebuild stop or stopduplexing request cannot be processed once the rebuild has entered the Cleanup phase. The rebuild process cannot be stopped. **Action:** Do not attempt to stop the rebuild or duplex rebuild process once the rebuild has entered the Cleanup phase.
C	xxxx0C3F	**Equate Symbol:** IXLRSNCODECONNNOTDEFINED **Meaning:** Environmental error. The connection making the rebuild complete or duplex complete request is not defined. **Action:** Ensure that the IXLREBLD REQUEST=COMPLETE or REQUEST=DUPLEXCOMPLETE is invoked by an active connector to the structure.
C	xxxx0C40	**Equate Symbol:** IXLRSNCODECONNNOTACTIVE **Meaning:** Environmental error. The connection indicating rebuild complete or duplex complete is not active. **Action:** Ensure that the IXLREBLD REQUEST=COMPLETE or REQUEST=DUPLEXCOMPLETE is invoked by an active connector to the structure.
C	xxxx0C41	**Equate Symbol:** IXLRSNCODEUNEXPECTEDRESPONSE **Meaning;** Environmental error. The system did not expect a rebuild complete or duplex complete request from the connection. **Action:** Examine the rebuild protocol in use by the application to ensure that the sequence of rebuild events/actions is correct.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C46	**Equate Symbol:** IXLRSNCODEREBUILDCOMPLETE **Meaning:** Environmental error. The system did not expect a rebuild complete request at this stage of the rebuild process. **Action:** Examine the rebuild protocol in use by the application to ensure that the sequence of rebuild events/actions is correct.
C	xxxx0C4A	**Equate Symbol:** IXLRSNCODEREBUILDNOTPERMITTED **Meaning:** Environmental error. At least one active connection specified the ALLOWREBLD=NO option on IXLCONN for the structure. This prevents structure rebuild or duplexing rebuild from being initiated for the structure. **Action:** Examine the rebuild protocol to determine why structure rebuild or duplexing rebuild is not allowed for this structure.
C	xxxx0C67	**Equate Symbol:** IXLRSNCODEREBLDNOOTHER **Meaning:** Environmental error. LOCATION=OTHER either was specified on the rebuild request or was defaulted to for STARTREASON=LOSSCONN or REQUEST=STARTDUPLEX rebuild requests. When duplexing is stopped by the operator and DUPLEX(ENABLED) is specified in the active policy for the structure, the subsequent duplexing rebuild request initiated due to DUPLEX(ENABLED) will both avoid the coupling facility in which the current structure is allocated and the coupling facility in which the previous instance of the structure was allocated when the duplexing rebuild was stopped. No other coupling facility exists in the active (or pending) policy preference list for the structure. **Action:** Ensure that the CFRM policy is set up with more than one coupling facility in the preference list for the structure.
C	xxxx0C6A	**Equate Symbol:** IXLRSNCODEREBLDNOOTHERCONN **Meaning:** Environmental error. No coupling facility in the preference list provided better connectivity than the current facility for this LOSSCONN rebuild. The rebuild was not started to avoid a further degradation in connectivity for the application. When STARTREASON=LOSSCONN is specified, the system ignores the LESSCONNACTION specification and processes the request as if LESSCONNACTION=TERMINATE were specified. **Action:** Either disconnect from the structure or reattempt to rebuild the structure by reissuing the IXLREBLD macro with a different STARTREASON.
C	xxxx0C6B	**Equate Symbol:** IXLRSNCODEREBLDINSUFFCONN **Meaning:** Environmental error. No coupling facility in the preference list provided better or equivalent connectivity than the current facility. The rebuild was not started to avoid a further degradation in connectivity for the application. **Action:** If you want the rebuild to occur despite a degradation in connectivity, use the LESSCONNACTION=CONTINUE option.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C6F	**Equate Symbol:** IXLRSNCODEDUPLEXNOTPERMITTED **Meaning:** Environmental error. The structure does not support duplexing rebuild for one of the following reasons: • The ALLOWDUPREBLD=NO option on IXLCONN was specified or defaulted to by at least one active or failed-persistent connection. • DUPLEX(DISABLED) was specified or defaulted to in the CFRM active policy for this structure. • A user-managed duplexing rebuild was required, and user-managed duplexing rebuilds are not supported for the type of structure. • There are connections pending reconciliation into the CFRM policy. • A system-managed duplexing rebuild is not supported when a CFRM policy change is pending for the structure. **Action:** Rebuild the structure first to cause the pending policy change to take effect.
C	xxxx0C70	**Equate Symbol:** IXLRSNCODEWRONGBUILDTYPE **Meaning:** Environmental error. Either of the following conditions exists: • IXLREBLD REQUEST=STOP was requested and a duplexing rebuild is in progress • IXLREBLD REQUEST=STOPDUPLEX was requested and a structure rebuild is in progress. **Action:** Examine the rebuild protocol in use by the application to ensure that the sequesnce of rebuild or duplexing events and actions is correct.
C	xxxx0C71	**Equate Symbol:** IXLRSNCODENOTDUPLEXESTAB **Meaning:** Environmental error. An IXLREBLD REQUEST=STOP to switch to the new structure was requested and the rebuild process is not yet in the Duplex Established phase. A stop to switch to the new structure cannot be accepted until the rebuild reaches the Duplex Established phase. **Action:** Examine the duplexing protocol in use by the application to ensure that the sequence of duplexing events and actions is correct.
C	xxxx0C72	**Equate Symbol:** IXLRSNCODEDUPLEXCOMPLETE **Meaning:** Environmental error. An IXLREBLD REQUEST=DUPLEXCOMPLETE request was not expected at this time. Either switch is not in progress or the connector has not established duplexing yet. if the latter, the connector must either establish duplexing or disconnect, allowing switch processing to proceed. **Action:** Enamine the duplexing protocol in use by the application to ensure that the sequence of duplexing events and actions is correct.
C	xxxx0C73	**Equate Symbol:** IXLRSNCODESTRFAILED **Meaning:** Environmental error. The rebuild request has been rejected because the structure has failed. A duplexing rebuild request will be rejected if the structure has failed. A structure rebuild request will be permitted. **Action:** Start a structure rebuild for the structure.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C74	**Equate Symbol:** IXLRSNCODESTOPPINGDIRECTION **Meaning:** Environmental error. The IXLREBLD REQUEST=STOPDUPLEX request was not processed because a request to stop the structure duplexing has already been initiated for the structure name in the other direction. Either KEEP=OLD was requested and a request to stop with KEEP=NEW is in progress or KEEP=NEW was requested and a request to stop with KEEP=OLD is in progress. **Action:** Examine the duplexing protocol in use by the application to ensure that the sequence of duplexing events and actions is correct.
C	xxxx0C75	**Equate Symbol:** IXLRSNCODEDUPLEXNOTFEASIBLE **Meaning:** The IXLREBLD START DUPLEX request was not processed because XES determined that allocation of the rebuild new structure would not be feasible. A duplexing rebuild might not be feasible because: • There is no coupling facility in the preference list that has CF-to-CF connectivity to the coupling facility in which the primary structure is allocated. • A loss of coupling facility connectivity would occur for active connectors to the structure. **Action:** Depending on the reason that the allocation of the rebuild new structure was not considered feasible, put actions in place to correct the situation.
C	xxxx0C80	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFINPROGRESS **Meaning:** Environmental error. An IXLREBLD REQUEST=START,POPULATECF=*cfname* was attempted while a previous request was already in progress. The current request is not processed. **Action:** Examine the protocol in use by the application to ensure that the sequence of events and actions when populating a coupling facility is correct.
C	xxxx0C81	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFNOTINPROGRESS **Meaning:** Environmental error. An IXLREBLD STOP,POPULATECF=*cfname* was attempted. However, there is no currently active POPULATECF request in progress for the specified coupling facility. The request is not processed. **Action:** Examine the protocol in use by the application to ensure that the sequence of events and actions when populating a coupling facility is correct.
C	xxxx0C83	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFNOSTRUCTS **Meaning:** Environmental error. An IXLREBLD REQUEST=START,POPULATECF=*cfname* was attempted. No structures were selected for the request. The request is not processed. **Action:** Examine the protocol in use by the application to ensure that the sequence of events and actions when populating a coupling facility is correct.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
C	xxxx0C84	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFFAILED **Meaning:** Environmental error. An IXLREBLD REQUEST=START,POPULATECF=*cfname* was attempted. The specified coupling facility has failed. The request is not processed. **Action:** Resubmit the request specifying a coupling facility that is available in the CFRM active policy or change the policy to reinstate the coupling facility.
C	xxxx0C85	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFINCLEANUP **Meaning:** Environmental error. An IXLREBLD REQUEST=START,POPULATECF=*cfname* was attempted. The specified coupling facility is in cleanup processing. The request is not processed. **Action:** Resubmit the request after the coupling facility has completed cleanup.
C	xxxx0C86	**Equate Symbol:** IXLRSNCODEREBUILDPOPCFDELETEPENDING **Meaning:** Environmental error. An IXLREBLD REQUEST=START,POPULATECF=*cfname* was attempted. The specified coupling facility is being deleted from the CFRM active policy. The request is not processed. **Action:** Resubmit the request specifying a coupling facility that is available in the CFRM active policy or change the policy to reinstate the coupling facility.
C	xxxx0C92	**Equate Symbol:** IXLRSNCODESYSMGDNOTSUPPORTEDSTR **Meaning:** An IXLREBLD REQUEST=START was attempted that needed system-managed processing (for example, rebuild). The system-managed process cannot be initiated for one of the following reasons: • The structure was not allocated in a coupling facility at or above the minimum CFLEVEL required for the current process. • The structure was not allocated by a system supporting system-managed processing. • The structure has connections that have not been reconciled into the CFRM active policy. • Structure cleanup is in progress for the structure (applicable to lock structures only). The request is not processed. **Action:** Determine the CFLEVEL of the coupling facility in which the structure is allocated through one of the following methods: using the IXLMG macro, issuing the DISPLAY XCF,STR and DISPLAY CF commands, or referencing the data returned in the CONA on the original connect to the structure. • If the CFLEVEL is too low, follow application protocols to shut down and allocate the structure in a coupling facility of suitable CFLEVEL. • If the CFLEVEL is sufficient, try the IXLREBLD request at a later time to allow CFRM policy reconciliation or lock structure cleanup to complete.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C93	**Equate Symbol:** IXLRSNCODESYSMGDSTRPREFLIST **Meaning:** An IXLREBLD REQUEST=START was attempted that needed system-managed processing (for example, rebuild). The system-managed process cannot be initiated because the preference list for the structure was unsuitable for one of the following reasons: • The preference list is empty. • The preference list contains no other coupling facility at the required CFLEVEL or higher. • The structure already exists in the only suitable coupling facility and this coupling facility could not be selected as the target for the system-managed process because no CFRM policy change is pending for this structure. The request is not processed. **Action:** The system programmer must update the structure's preference list to include at least two coupling facilities at the required CFLEVEL.
C	xxxx0C94	**Equate Symbol:** IXLRSNCODESYSMGDNOTSUPPORTEDCONN **Meaning:** An IXLREBLD REQUEST=START request was attempted and would have resulted in system-managed processing (for example, rebuild). The system-managed process could not be initiated because there is at least one active connection and all connections did not specify ALLOWAUTO=YES on IXLCONN. The request is not processed. **Action:** The application must support either user-managed or system-managed rebuild for the rebuild start request to succeed. If the application intends to support system-managed processing, all active connectors that specified or defaulted to IXLCONN ALLOWAUTO=NO must disconnect and, if desired, reconnect with ALLOWAUTO=YES, before the request can be processed successfully.
C	xxxx0C95	**Equate Symbol:** IXLRSNCODESYSMGDBADSTARTREASON **Meaning:** An IXLREBLD REQUEST=START invocation would have resulted in a system-managed rebuild. The rebuild could not be initiated because the request specified a STARTREASON of LOSSCONN or STRFAILURE, which are not valid reasons for starting a system-managed rebuild. The request is not processed. **Action:** Specify a STARTREASON other than LOSSCONN or STRFAILURE.
C	xxxx0C96	**Equate Symbol:** IXLRSNCODESYSMGDLOSSCONN **Meaning:** An IXLREBLD REQUEST=START invocation would have resulted in a system-managed rebuild. The rebuild could not be initiated because an active or failing connector does not have connectivity to the target structure. The request is not processed. **Action:** Try the start request again when all connectors who have lost connectivity to the structure have disconnected and become either failed-persistent or undefined.

Table 79. Return and Reason Codes for the IXLREBLD Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C97	**Equate Symbol:** IXLRSNCODESYSMGDCOMPLETENOTPERMITTED **Meaning:** An IXLREBLD REQUEST=COMPLETE or REQUEST=DUPLEXCOMPLETE was issued for a structure that is undergoing system-managed processing (for example, rebuild). The request is not processed. **Action:** Do not issue IXLREBLD REQUEST=COMPLETE or REQUEST=DUPLEXCOMPLETE against a structure that is undergoing system-managed processing.
C	xxxx0C99	**Equate Symbol:** IXLRSNCODESYSMGDNOTSUPPORTEDCDS **Meaning:** An IXLREBLD REQUEST=START request was attempted which needed system-managed processing (for example, rebuild). The system-managed process could not be initiated because the CFRM couple data set was not formatted at the minimum required level. The request is not processed. **Action:** Format and activate a CFRM couple data set that supports the requested system-managed process. For rebuild, specify ITEM NAME(SMREBLD) NUMBER(1) when formatting.
C	xxxx0C9B	**Equate Symbol:** IXLRSNCODESYSMGDNOHISTORY **Meaning:** A request to initiate a duplexing rebuild was attempted which needed system-managed processing. The system-managed duplexing rebuild cannot be initiated because there are no connections to the structure and the structure has not previously been duplexed using system-managed processing. The request is not processed. **Action:** Start a connector that supports system-managed duplexing rebuild, and try again.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Equate Symbol:** IXLRSNCODESYSMGDXCFERROR **Meaning:** Failure in XES processing. The state of the resource request is unpredictable. **Action:** Save the reason code information and contact the IBM support center.

IXLRT — Lock Structure Record Data Processing

Description

The IXLRT service enables you to perform recovery procedures if a connection to a lock structure fails. The IXLRT service is the recovery interface to obtain and clean up recording information in a lock structure.

The following services are available:
- Create a record data entry and write data to the entry.
- Read the entire set of record data entries in the lock structure.
- Read the entire set of record data entries associated with a connected user.
- Read a single record data entry by entry identifier.
- Delete all record data entries identified by a list of entry identifiers.
- Delete the entire set of record data entries associated with a connected user.
- Delete a single record data entry by entry identifier.
- Update a record data entry by entry identifier.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or PKM allowing key 0 - 7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued.
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL)

Programming Requirements

Before using the IXLRT service, you must identify your connection to the system through the IXLCONN service. On successful completion of the IXLCONN service, you receive a sysplex-wide unique CONNECT token. This token identifies your connection to the lock structure. The connect token must be specified on every IXLRT request to ensure that you are allowed access to the designated record data structure.

The caller's parameter list must be addressable from the unit of work issuing the request.

The IXLYMRTD macro provides the format of the area that the DATAREA points to. Include that macro in your program.

The IXLYRTAA mapping macro provides the format of the area that the ANSAREA points to. Include that macro in your program.

Restrictions and Limitations

The system does not serialize the record data entry. Therefore, unless the caller provides serialization to prevent entries from being created while IXLRT reads or deletes entries, it is possible that not all record data entries will be returned to the requestor.

Input Register Information

Before issuing the IXLRT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLRT macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code if GPR15 return code is nonzero
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller of the IXLRT macro, the access registers (ARs) contain:

Register	Contents
0-1	Used as a work register by the system
2-13	Unchanged
14-15	Used as a work register by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Performance Implications

None.

Understanding IXLRT Version Support

The IXLRT macro supports versions 0 through 4.

- Keywords not specifically noted here are supported by all versions starting with version 0 and higher of the IXLRT macro.
- The following keyword is supported by all versions starting with version 1 and higher of the IXLRT macro.

FASTRESTOKEN

- The following keyword is supported by all versions starting with version 2 and higher of the IXLRT macro.

RDATATYPE

- The following keyword is supported by all versions starting with version 3 and higher of the IXLRT macro.

EXTRESTOKEN

- The following keywords are supported by all versions starting with version 4 and higher of the IXLRT macro.

MRTDLEVEL

OUTRDATATYPE

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLRT macro is as follows:

main diagram

```
►►──IXLRT──ƀ──CONTOKEN=contoken──────────────────────────────────────────────────────────►

►─,REQUEST=──┬─CREATENTRY──┤ parameters-1 ├──────────────────────────┬─,ANSAREA=ansarea──────►
             ├─READALL──┤ parameters-2 ├─────────────────────────────┤
             ├─READBYCONN──┤ parameters-3 ├──────────────────────────┤
             ├─READENTRY──┤ parameters-4 ├───────────────────────────┤
             ├─DELETENTRYLST──┤ parameters-5 ├───────────────────────┤
             │             ┌─,VERCONNAME=NO_VERCONNAME─┐             │
             ├─DELETENTRY──┼───────────────────────────┼─,ENTRYID=entryid─┤
             │             └─,VERCONNAME=verconname────┘             │
             ├─DELETEBYCONN──┤ parameters-6 ├────────────────────────┤
             └─UPDATENTRY──┤ parameters-7 ├──────────────────────────┘

                                                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►─,ANSLEN=anslen──┬──────────────────┬──┬──────────────────┬───┼───────────────────────────┼──►
                  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘   ├─,PLISTVER=MAX─────────────┤
                                                               └─,PLISTVER=plistver────────┘

  ┌─,MF=S─────────────────────────────┐
►─┼───────────────────────────────────┼──────────────────────────────────────────────────►◄
  │                  ┌─,0D───────┐    │
  ├─,MF=(L,mfctrl────┼───────────┼─)──┤
  │                  └─,mfattr───┘    │
  │                  ┌─,COMPLETE─┐    │
  └─,MF=(E,mfctrl────┼───────────┼─)──┘
                     └─,COMPLETE─┘
```

parameters-1

```
     ┌─,TCONNAME=NO_TCONNAME─┐                                   ┌─,RDATATYPE=0──────────┐
►►───┼───────────────────────┼─,ENTRYID=entryid──,RDATAVAL=rdataval──┼───────────────────────┼──►◄
     └─,TCONNAME=tconname────┘                                   └─,RDATATYPE=rdatatype──┘
```

parameters-2

```
                                                      ┌─,MRTDLEVEL=0──────────┐
►►───┬─,RESTOKEN=restoken─────────┬─,DATAREA=datarea──┼───────────────────────┼──────────►
     └─,EXTRESTOKEN=extrestoken───┘                   └─,MRTDLEVEL=mrtdlevel──┘

  ┌─,RDATATYPE=NO_RDATATYPE─┐
►─┼─────────────────────────┼────────────────────────────────────────────────────────────►◄
  └─,RDATATYPE=rdatatype────┘
```

parameters-3

```
   ┌─,TCONNAME=NO_TCONNAME─┐   ┌─,RDATATYPE=NO_RDATATYPE─┐
►►─┼───────────────────────┼───┼─────────────────────────┼──────────────────────►
   └─,TCONNAME=tconname────┘   └─,RDATATYPE=rdatatype────┘

  ┌─,FASTPATH=NO──┬─,RESTOKEN=restoken───────┬─┐                    ┌─,MRTDLEVEL=0─────────┐
  │               └─,EXTRESTOKEN=extrestoken─┘ │                    │                      │
►─┼────────────────────────────────────────────┼─,DATAREA=datarea───┼──────────────────────┼────►◄
  └─,FASTPATH=YES,FASTRESTOKEN=fastrestoken────┘                    └─,MRTDLEVEL=mrtdlevel─┘
```

parameters-4

```
   ┌─,VERCONNAME=NO_VERCONNAME─┐
►►─┼───────────────────────────┼─,ENTRYID=entryid──,RDATAVAL=rdataval──────────►
   └─,VERCONNAME=verconname────┘

►─┬──────────────────────────────┬──────────────────────────────────────────────►◄
  └─,OUTRDATATYPE=outrdatatype───┘
```

parameters-5

```
                             ┌─,FIRSTELEM=1─────────┐
►►──,ENTRYIDLIST=entryidlist─┼──────────────────────┼─,LASTELEM=lastelem───────►◄
                             └─,FIRSTELEM=firstelem─┘
```

parameters-6

```
   ┌─,TCONNAME=NO_TCONNAME─┐                                ┌─,RDATATYPE=NO_RDATATYPE─┐
►►─┼───────────────────────┼──┬─,RESTOKEN=restoken───────┬──┼─────────────────────────┼──────►◄
   └─,TCONNAME=tconname────┘  └─,EXTRESTOKEN=extrestoken─┘  └─,RDATATYPE=rdatatype────┘
```

parameters-7

```
   ┌─,VERCONNAME=NO_VERCONNAME─┐                                      ┌─,RDATATYPE=0─────────┐
►►─┼───────────────────────────┼─,ENTRYID=entryid──,RDATAVAL=rdataval─┼──────────────────────┼──►◄
   └─,VERCONNAME=verconname────┘                                      └─,RDATATYPE=rdatatype─┘
```

Note: FASTPATH=NO is the default and does not need to be coded. However, if you select this parameter, you must code either RESTOKEN=*restoken* or EXTRESTOKEN=*extrestoken*.

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ANSAREA=*ansarea*

Use this output area to contain information about the request. The storage area is mapped by the IXLYRTAA macro and must be addressable in the caller's primary address space.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an output area is to contain information about the request.

,ANSLEN=*anslen*

Use this input area to specify the length of the ANSAREA storage area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword input field that specifies the length of ANSAREA.

CONTOKEN=*contoken*

Use this input parameter to specify the connect token of the requestor. The token is the connect token that was returned to the requestor by the IXLCONN service. CONTOKEN uniquely identifies the user's connection to a lock structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character input field that contains the connect token returned in the answer area by the IXLCONN service.

,DATAREA=*datarea*

Use this output area to contain the data returned by a READALL or READBYCONN request. This area must be 4096 bytes long and is mapped by the IXLYMRTD macro.

Upon successful completion of a READALL or READBYCONN request, DATAREA contains, starting at offset zero, an array of records containing 64 bytes of record data, the entry identifier (ENTRYID) for the element that was successfully read, and the connection identifier (CONID) of the connected user who is associated with each element that was successfully read. Additional information may also be provided, based on the MRTDLEVEL requested. The number of record data entries returned is indicated in the answer area specified by ANSAREA, mapped by IXLYRTAA.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4096-byte output area to contain the data returned by a READALL or READBYCONN request.

,ENTRYID=*entryid*

Use this input/output parameter to specify the identifier assigned to the record data entry.

For a CREATENTRY request, this identifier is returned by IXLRT and must be used on all subsequent requests to access the record data entry.

For READENTRY and DELETENTRY requests, this identifier specifies the unique entry identifier assigned to the record data entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character input/output field that contains the entry identifier.

,ENTRYIDLIST=*entryidlist*

Use this input parameter to identify an area that contains a list of record data entry identifiers to be processed by the DELETENTRYLST request. This list must be aligned on a page boundary and be 4096 bytes long. The ENTRYIDLIST must be formatted into 12-byte elements starting at offset zero. Each record data entry to be processed consists of a 12-byte identifier.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 4096-byte input area to contain a list of record data entry identifiers to be processed by a DELETENTRYLIST request.

,EXTRESTOKEN=*extrestoken*

Use this input/output parameter to specify an extended restart token that can be used to resume processing of all READALL, READBYCONN, and DELETEBYCONN requests. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token (RESTOKEN).

You must provide an initial value of zero in this field. The system then initializes the field with control information relevant to the READALL, READBYCONN, or DELETEBYCONN request. On each subsequent request, the system uses this information. You must not modify the contents of this field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character input/output area to contain an extended restart token.

,FASTPATH=NO
,FASTPATH=YES

Use this input parameter to indicate that the system is to process this READBYCONN request using an optimized access method when accessing the record data.

NO
The system is not to use the optimized access method when accessing the record data.

YES
The system is to use the optimized access method when accessing the record data. Use FASTPATH=YES only when the caller has serialization that ensures that the record data entries belonging to the target connector will remain unchanged throughout the process.

,FASTRESTOKEN=*fastrestoken*
Use this input/output parameter to specify a 12-character field that must be passed on READBYCONN FASTPATH=YES requests.

You must provide an initial value of zero in the field. The system then initializes the field with control information. On each subsequent READBYCONN FASTPATH=YES request, the system uses this information. You must not modify the contents of this field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character input/output field to contain control information.

,FIRSTELEM=1
,FIRSTELEM=*firstelem*
Use this input parameter to specify the index of the first ENTRYIDLIST element to be processed. The value specified must be in the range of 1 to 341 inclusive.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword input field to contain the index of the first ENTRYIDLIST element to be processed.

,LASTELEM=*lastelem*
Use this input parameter to specify the index of the last ENTRYIDLIST element to be processed. The value must be in the range of 1 to 341 inclusive and must be greater than, or equal to, the value of FIRSTELEM.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword input field to contain the index of the last ENTRYIDLIST element to be processed.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,MRTDLEVEL=0
,MRTDLEVEL=*mrtdlevel*

Use this input parameter to specify the level of MRTD data that is to be returned in the area specified by DATAREA. Valid values are 0 and 1.

- A value of 0 indicates that the entries returned will be mapped by the MRTD mapping in IXLYMRTD.
- A value of 1 indicates the entries returned will be mapped by the MRTD1 mapping in IXLYMRTD.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the one-byte area containing the level of the IXLYMRTD record mappings.

,OUTRDATATYPE=*outrdatatype*

Use this output parameter to contain the record data type (RDATATYPE) value associated with the record data element that was returned for a READENTRY request.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character field that will contain the returned record data element's record data type.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*

Use this input parameter to specify the version of the macro. See "Understanding IXLRT Version Support" on page 1192 for a description of the options available with PLISTVER.

,RDATATYPE=NO_RDATATYPE
,RDATATYPE=0
,RDATATYPE=*rdatatype*

Use this input paramter to specify the type of record data in the lock structure.

For a CREATENTRY request, RDATATYPE specifies the record data type that is to be assigned to the created record data entry. If a nonzero record data type value is not specified, a record data type of zero will be assigned to the entry. Note that record data that is created by the IXLLOCK service in conjuntion with a held lock resource has a record data type of zero. DEFAULT: 0.

For a READALL, READBYCONN, and DELETEBYCONN request, RDATATYPE specifies the record data type to be used as a filter for processing the record data entries. When specified, only those record data entries whose record data type matches the specified value will be selected for processing. Note that record data that is created by the IXLLOCK service in conjunction with a held lock resource has a record data type of zero. DEFAULT: NO_RDATATYPE.

For an UPDATENTRY request, RDATATYPE specifies the record data type to be assigned to the updated record data entry. The record data type can be the same as or different from that current record data type for the entry. If a nonzero record data type value is not specified, a record data type of zero will be assigned to the entry. Note that record data that is created by the IXLLOCK service in conjunction with a held lock resource has a record data type of zero. DEFAULT: 0

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an 8-character field that contains the record data type.

,REQUEST=CREATENTRY

,REQUEST=READALL
,REQUEST=READBYCONN
,REQUEST=READENTRY
,REQUEST=DELETENTRYLST
,REQUEST=DELETENTRY
,REQUEST=DELETEBYCONN
,REQUEST=UPDATENTRY

Use this input parameter to specify the function requested.

CREATENTRY

Create a record data entry in the lock structure to which the caller is connected and write data to that entry.

When the request completes successfully, the unique entry identifier assigned by XES, the number of entries associated with the target connector, and the total number of in-use entries in the record data entry are returned in the answer area specified by ANSAREA.

READALL

Read the next group of record data entries associated with the lock structure to which the caller is connected.

When the request completes successfully, the number of entries for which record data was read is returned in the answer area specified by ANSAREA.

A READALL request may complete prematurely. If so, indicative return and reason codes are provided, the number of record data entries read is provided in the answer area, and a token is returned in the location specified by the RESTOKEN or EXTRESTOKEN keyword. This token may be specified as input on a subsequent READALL request to resume processing with the appropriate record data entry.

READBYCONN

Read the next group of record data entries associated with the specified connection name.

When the request completes successfully, the number of entries for which record data was read is returned in the answer area specified by ANSAREA.

A READBYCONN request may complete prematurely. If so, indicative return and reason codes are provided, the number of record data entries read is provided in the answer area, and a token is returned in the location specified by the RESTOKEN or EXTRESTOKEN keyword. This token may be specified as input on a subsequent READBYCONN request to resume processing with the appropriate record data entry.

READENTRY

Read a particular record data entry by entry identifier.

When the request completes successfully, the number of entries associated with the target connector and the total number of in-use record data entries are returned in the answer area specified by ANSAREA.

DELETENTRYLST

Delete all record data entries specified by a list of entry identifiers. This list of entry identifiers is specified by the ENTRYIDLIST keyword.

When the request completes successfully, the number of entries deleted for this request is returned in the answer area specified by ANSAREA.

A DELETENTRYLST request may complete prematurely. If so, indicative return and reason codes are provided, the number of record data entries deleted, and the index of the next entry to be deleted are provided in the ANSAREA. To continue deleting the remaining elements in the ENTRYIDLIST, the DELETENTRYLST request can be reissued with the FIRSTELEM keyword updated to indicate the new starting point in the element list.

If any entry specified in the ENTRYIDLIST does not exist, the processing is halted and the index of the offending element in the ENTRYIDLIST is also returned in the answer area. When this occurs, all specified elements preceding the offending element have been processed. All succeeding elements have not been processed. To continue deleting the remaining elements in the ENTRYIDLIST, the DELETENTRYLST request can be reissued with the FIRSTELEM keyword updated to indicate the new starting point in the element list.

DELETENTRY
Delete an existing record data entry by entry identifier.

When the request completes successfully, the remaining number of entries associated with the target connector and the remaining total number of in-use record data entries are returned in the answer area specified by ANSAREA.

DELETEBYCONN
Delete the next group of record data entries associated with the specified connection name.

When the request completes successfully, the number of entries for which record data was deleted is returned in the answer area specified by ANSAREA.

A DELETEBYCONN request may complete prematurely. If so, indicative return and reason codes are provided, the number of record data entries deleted is provided in the answer area, and a token is returned in the location specified by the RESTOKEN or EXTRESTOKEN keyword. This token may be specified as input on a subsequent DELETEBYCONN request to resume processing with the appropriate record data entry.

UPDATENTRY
Update an existing record data entry by entry identifier.

When the request completes successfully, the number of entries associated with the connector with whom the updated entry was associated and the total number of in-use record data entries are returned in the answer area specified by ANSAREA.

,RDATAVAL=*rdataval*
Use this input/output parameter to specify 64 bytes of user-defined data.

For a CREATENTRY request, the field contains user-defined data to be written to the record data entry.

For a READENTRY request, the field will contain the data returned from the record data entry.

For an UPDATENTRY request, the field contains the data with which to update the record data entry.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 64-character input/output field to contain the user-defined data for a record data entry.

,RESTOKEN=*restoken*
Use this input/output parameter to specify a containing a standard restart token that must be passed on READALL, READBYCONN FASTPATH=NO, and DELETEBYCONN requests. Requestors that specify or default to IXLCONN ALLOWAUTO=NO must use the standard restart token. Requestors that specify IXLCONN ALLOWAUTO=YES must use the extended restart token (EXTRESTOKEN).

You must provide an initial value of zero in the field. The system then initializes the field with control information relevant to the READ or DELETE request. On each subsequent READ or DELETE request, the system uses this information. You must not modify the contents of this field.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of an eight-character input/output field to contain control information.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,TCONNAME=NO_TCONNAME
,TCONNAME=*tconname*
Use this input parameter to specify the target connection name (CONNAME) of the connected user. If a target connection name is not specified, the record data is associated with the connected user identified by the CONTOKEN keyword.

For a CREATENTRY request, the record data entry that is to be created will be associated with the connected user identified by the TCONNAME keyword.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character input field that contains the target connection name.

,VERCONNAME=NO_VERCONNAME
,VERCONNAME=*verconname*
Use this input parameter to specify the name of a connected user to be verified as the connector associated with the record data entry that is to be read. if the connector indicated by the VERCONNAME is not associated with the entry specified by ENTRYID, the IXLRT request fails.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character input field to contain the connection name to be verified.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the IXLRT macro returns control to your program:
- GPR 15 (and *retcode*, you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code if applicable.

Macro IXLRT provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 80. Return and Reason Codes for the IXLRT Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** IXLRT request successful. **Action:** None.

Table 80. Return and Reason Codes for the IXLRT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
4	xxxx0409	**Equate Symbol:** IXLRSNCODETIMEOUT **Meaning:** • A READALL, READBYCONN, DELETEBYCONN, or DELETENTRYLST request has completed prematurely due to a model-dependent time-out condition. The number of reliable record data entries that has been read or deleted has been returned in the answer area. • All other request types: Not applicable. **Action:** Respecify the request by issuing it with the RESTOKEN or EXTRESTOKEN returned for READALL, READBYCONN, and DELETEBYCONN, or by adjusting FIRSTELEM on a DELETENTRYLST request.
4	xxxx040F	**Equate Symbol:** IXLRSNCODEBUFFERFULL **Meaning:** • A READALL or READBYCONN request has completed prematurely due to a buffer full condition. The number of reliable record data entries that have been read has been returned in the answer area. • All other request types: Not applicable. **Action:** Process the data returned in DATAREA and then reissue the request specifying RESTOKEN or EXTRESTOKEN and an empty DATAREA.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** Program error. The parameter list is either not addressable or not accessible. **Action:** Verify that the parameter list address is valid.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the parameter list is not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS on which your program is running.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** Program error. Caller is not enabled. **Action:** Verify that the program is enabled for I/O and external interrupts.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Invalid CONTOKEN specified. The contoken is invalid for one of the following reasons: disconnect has occurred, EOT of the connector's task, input contoken is not the contoken returned from IXLCONN, the request was issued outside the connector's address space, or the contoken has been invalidated for rebuild. **Action:** Verify that the CONTOKEN value specified is valid and for the correct structure.

Table 80. Return and Reason Codes for the IXLRT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx080B	**Equate Symbol:** IXLRSNCODEBADCONNAME **Meaning:** Program error. The TCONNAME or VERCONNAME specified was not valid. **Action:** Ensure that the TCONNAME or VERCONNAMEwas specified correctly.
8	xxxx080E	**Equate Symbol:** IXLRSNCODEBADAREA **Meaning:** Program error. The answer area specified by ANSAREA is either not addressable or not accessible. **Action:** Ensure that the address specified for ANSAREA is valid.
8	xxxx0816	**Equate Symbol:** IXLRSNCODENORTEXISTS **Meaning:** Program error. There are no record data entries allocated. **Action:** Ensure that RECORD=YES was specified on the IXLCONN macro to provide recording.
8	xxxx0818	**Equate Symbol:** IXLRSNCODENOTLOCKSTR **Meaning:** Program error. The connection specified by CONTOKEN does not represent a lock structure. **Action:** Verify that the CONTOKEN is specified correctly and is for the correct structure.
8	xxxx081A	**Equate Symbol:** IXLRSNCODENORTENTRY **Meaning:** Program error. A request to read, update, or delete a record data entry found no such entry allocated. **Action:** Verify that the record data entry specified is correctly identified.
8	xxxx082B	**Equate Symbol:** IXLRSNCODEBADIDINDEX **Meaning:** Program error. A DELETEENTRYLIST request had an invalid index specified, either by FIRSTELEM or LASTELEM, for the first or last element in the element list. The RTAADELCNT and RTAAFAILINDEX fields in the RTAA will contain the count of elements deleted and the index of the failing entry, respectively. **Action:** Update FIRSTELEM to point past the failing entry and re-issue the request.
8	xxxx0835	**Equate Symbol:** IXLRSNCODEBADDATAADDR **Meaning:** Program error. The storage area specified by DATAREA or ENTRYIDLIST is not addressable. **Action:** Ensure that the address specified for DATAREA or ENTRYIDLIST is valid.
8	xxxx083D	**Equate Symbol:** IXLRSNCODEBADANSLEN **Meaning:** Program error. The length of the answer area, as specified by ANSLEN, is not sufficient for providing answer area information. **Action:** Determine the length of the answer area and correct the value specified in ANSLEN.

Table 80. Return and Reason Codes for the IXLRT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0849	**Equate Symbol:** IXLRSNCODEBADRESTOKEN **Meaning:** Program error. • A READALL, READBYCONN, or DELETEBYCONN request specified a restart token that was not valid. Possible causes are: – The specified token does not correspond to a previous prematurely-completed request. – The user specified RESTOKEN when EXTRESTOKEN was required. – The user specified EXTRESTOKEN when RESTOKEN was required. • All other request types: Not applicable. **Action:** Ensure that you specified the correct restart token and that you have not modified it.
8	xxxx0855	**Equate Symbol:** IXLRSNCODEENTRIESCHANGED **Meaning:** Program error. The record table entry that was represented by the FASTRESTOKEN was deleted or reacquired between IXLRT REQUEST=READBYCONN FASTPATH=YES requests. **Action:** The request cannot be processed. Reset the value of FASTRESTOKEN to zero and restart the IXLRT process from the beginning.
8	xxxx0887	**Equate Symbol:** IXLRSNCODEBADEXTRESTOKEN **Meaning:** Program error. • A READALL, READBYCONN, or DELETEBYCONN request specified an extended restart token that was not valid. The specified token refers to an older instance of the target structure. • All other request types: Not applicable. **Action:** Reset the value of EXTRESTOKEN to zero and resubmit the IXLRT request.
8	xxxx08A8	**Equate Symbol:** IXLRSNCODEBADMRTDLEVEL **Meaning:**Program error. The value specified for MRTDLEVEL was not valid. **Action:** Correct your program so that it specifies a valid value for MRTDLEVEL.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** Environmental error. No connectivity to structure containing record data. This may occur due to operator commands such as VARY PATH OFFLINE or CONFIG CHP OFFLINE or hardware errors such as coupling facility or path failures. The contoken will be invalidated. **Action:** Disconnect from the structure or rebuild.

Table 80. Return and Reason Codes for the IXLRT Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C0B	**Equate Symbol:** IXLRSNCODERTFULL **Meaning:** • The record portion of the lock structure is full and cannot accommodate the CREATENTRY request. • All other request types: Not applicable. **Action:** Rebuild or alter the structure to allow for more record data entries.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Environmental error. Prior to the completion of the request, the lock structure failed. **Action:** Attempt to rebuild the structure using IXLREBLD or disconnect from the structure using IXLDISC.
C	FFFFFFFF	**Meaning:** XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES function is not present. **Action:** Re-IPL the system, or follow your particular management protocol.
10	xxxx10xx	**Meaning:** Failure in XES processing. The state of the resource request is unpredictable. **Action:** Save the reason code information and contact the IBM support center.

IXLSYNCH — Synchronous Update to a Lock Structure

Description

The IXLSYNCH service enables a connected user to modify information in the notify exit parameter list (NEPL). The IXLSYNCH service is called from the notify exit to provide a synchronous update of the state and/or user data associated with a resource. When the notify exit returns control to the system, the updated data can be used to resolve resource contention.

Environment

The requirements for the caller are:

Minimum Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	SRB
Cross memory mode:	Any PASN, any HASN, any SASN. The primary address space must be the same as the primary address space at the time the connection service (IXLCONN) was issued.
AMODE:	31-bit
ASC mode:	Primary or Access Register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Must be in the primary address space

Programming Requirements

The IXLSYNCH service is called from the notify exit to update information in the notify exit parameter list (NEPL). The IXLSYNCH service is the only method of successfully updating this information before the notify exit returns control to the system. If the connected user makes changes to the NEPL and does not invoke the IXLSYNCH service, the changes to the NEPL are ignored.

The IXLYNEPL macro provides the format of notify exit parameter list. Include that macro in your program.

Restrictions and Limitations

The IXLSYNCH service is supported only when issued out of the notify exit.

Input Register Information

Before issuing the IXLSYNCH macro, the caller does not have to place any information in any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLSYNCH macro, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code if GPR15 return code is nonzero
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller of the *IXLSYNCH macro*, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as a work register by the system
2-13	Unchanged

14-15 Used as a work register by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro, and restore them after the system returns control.

Performance Implications

None.

Understanding IXLSYNCH Version Support

The IXLSYNCH macro supports version 0 keywords and functions.

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLSYNCH macro is as follows:

main diagram

```
►►──IXLSYNCH──b──NEPL=nepl──┬───────────────────┬──┬───────────────────┬──────────────────────────►
                            └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S────────────────────────────────┐
►──┼───────────────────────────┼───┼──────────────────────────────────────┼──────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤   │                    ┌─,0D──────┐      │
   └─,PLISTVER=plistver────────┘   ├─,MF=(L─,mfctrl─────┼──────────┼─)────┤
                                   │                    └─,mfattr──┘      │
                                   │                    ┌─,COMPLETE─┐     │
                                   └─,MF=(E─,mfctrl─────┼───────────┼─)───┘
                                                        └─,COMPLETE─┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,MF=S
,MF=(L,*mfctrl*)
,MF=(L,*mfctrl,mfattr*)
,MF=(L,*mfctrl*,0D)
,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

NEPL=*nepl*
Use this input parameter to identify the notify exit parameter list (IXLYNEPL). The parameter list must be the actual list passed and not a copy of the list.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list that was passed to the notify exit in register 1.

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=*plistver*
Use this input parameter to specify the version of the macro. See "Understanding IXLSYNCH Version Support" on page 1206 for a description of the options available with PLISTVER.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.)

Return and Reason Codes

When the IXLSYNCH macro returns control to your program,
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code as follows:

0 IXLRETCODEOK

4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 81. Return and Reason Codes for the IXLSYNCH Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
0	None.	**Meaning:** IXLSYNCH request successful. Requested changes are reflected in the corresponding system's local structure. **Action:** None
4	xxxx041E	**Equate Symbol:** IXLRSNCODESYNCHRTNOTDELETED **Meaning:** The resource was released through IXLSYNCH. However, a record data element could not be deleted. **Action:** None expected.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** The version number in the parameter list is not valid. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS that your program is running on.
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** IXLSYNCH request unsuccessful. The caller is not enabled. **Action:** Verify that the program is enabled for I/O and external interrupts.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** Invalid CONTOKEN specified. The contoken is invalid for one of the following reasons: disconnect has occurred, EOT of the connector's task, input contoken is not the contoken returned from IXLCONN, the request was issued outside the connector's address space, or the contoken has been invalidated for rebuild. **Action:** Verify that the CONTOKEN value specified is valid and for the correct structure.
8	xxxx0810	**Equate Symbol:** IXLRSNCODERESOURCENOTFOUND **Meaning:** IXLSYNCH request unsuccessful. Resource that caused the notify exit to be given control has been released by a previous invocation of IXLSYNCH. **Action:** None expected. If necessary, attempt to regain ownership of the resource.

Table 81. Return and Reason Codes for the IXLSYNCH Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0811	**Equate Symbol:** IXLRSNCODESYNCHBADSTATE **Meaning:** IXLSYNCH request unsuccessful. Attempt to change STATE to a value other than shared, exclusive, or free. **Action:** Valid STATE values are shared, exclusive, and free. See IXLYCON.
8	xxxx0815	**Equate Symbol:** IXLRSNCODEBADNEPL **Meaning:** Specified NEPL is not valid. Note that the NEPL is no longer valid upon return from the notify exit to which it was presented. **Action:** Ensure that the NEPL address is specified correctly. Also, verify that the protocol does not attempt to reacquire the NEPL after returning from the notify exit.
8	xxxx0816	**Equate Symbol:** IXLRSNCODENORTEXISTS **Meaning:** IXLSYNCH request unsuccessful. Record structure does not exist. **Action:** Ensure that RECORD=YES was specified on the IXLCONN macro to request recording.
C	xxxx0C06	**Equate Symbol:** IXLRSNCODENOCONN **Meaning:** No connectivity to lock structure. This may occur due to operator commands such as VARY PATH OFFLINE or CONFIG CHP OFFLINE or hardware errors such as coupling facility or path failures. The contoken will be invalidated. **Action:** Either disconnect from the structure or rebuild.
C	xxxx0C0B	**Equate Symbol:** IXLRSNCODERTFULL **Meaning:** Record structure full. **Action:** If your protocol allows, attempt to rebuild the lock structure so that additional record data might be available.
C	xxxx0C13	**Equate Symbol:** IXLRSNCODEREQPURGED **Meaning:** Environmental error. The request was purged prior to completion of the request. Possible reasons include: • The connector failed. • The connector disconnected. • The requestor failed. • The request was purged by IXLPURGE. • Requests were purged when the connector provided an IXLEERSP response for the Rebuild Stop or Rebuild Cleanup event. • The secondary address space was no longer valid. **Action:** None if this is expected. Otherwise, determine why the connector failed.
C	xxxx0C25	**Equate Symbol:** IXLRSNCODESTRFAILURE **Meaning:** Prior to completion of the request, the lock structure failed. **Action:** Attempt to rebuild the structure using IXLREBLD or disconnect from the structure using IXLDISC.

Table 81. Return and Reason Codes for the IXLSYNCH Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Locking functions are not available. This can occur because the coupling facility hardware necessary to provide locking function is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.
10	xxxx10xx	**Meaning:** Failure in XES processing. The state of the resource request is unpredictable. **Action:** Save the reason code information and contact the IBM support center.

IXLUSYNC — Synchronizing Processing for User-Defined Events

Description

Use the IXLUSYNC macro to synchronize the processing of user-defined events for multiple connectors to a given structure. IXLUSYNC allows users to establish “sync points” to indicate that all active connectors to a structure have confirmed that processing for an event is complete. Connectors are notified when a sync point is reached by means of the User Sync Point event being presented to their event exit.

You can use IXLUSYNC:

- To define a value to be associated with a user-defined event, the processing for which is to be synchronized among all active connectors.
- To confirm that the processing related to an event has been completed. When all active connectors have confirmed the event, the sync point is reached.
- To confirm that the processing associated with an event has completed *and* then to set a new event for which processing is to be synchronized.

You can also use IXLUSYNC to provide a response on behalf of a peer connector that has disconnected or failed prior to responding to a user-defined event.

Only one user-defined event can be set at a time for the active connectors to a structure.

Environment

The requirements for the caller are:

Authorization:	Supervisor state or PKM keys 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN; any SASN; any HASN. The primary address space must be equal to the requestor’s primary address space at the time of the connection.
AMODE:	31-bi
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held, with no enabled, unlocked task (EUT) FRRs established
Control parameters:	Must be in the primary address space or be in an address/data space that is addressable through a public entry on the caller’s dispatchable unit access list (DU-AL)

Restrictions

None.

Input Register Information

Before issuing the IXLUSYNC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLUSYNC macro, the general purpose registers (GPRs) contain:

Register	**Contents**
0	Reason code, if applicable, if GPR 15 return code is nonzero
1	Used as work register by the system
2-13	Unchanged

14	Used as work register by the system
15	Return code

When control returns to the caller of the IXLUSYNC macro, the access registers (ARs) contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Programming Requirements

If the program is in AR mode, issue the SYSSTATE ASCENV=AR macro before IXLUSYNC. SYSSTATE ASCENV=AR tells the system to generate code appropriate for AR mode.

Performance Implications

IXLUSYNC processing requires that all active connections reach a sync point before proceeding. The performance of IXLUSYNC depends on the environment, the number of connections to the structure, and the processing required to reach the sync point. Also, depending on the protocols of connections during IXLUSYNC, the services provided by those connections may be temporarily unavailable.

IXLUSYNC processing might involve referencing or updating the CFRM active policy to reflect the operation that has been requested. When invoking this macro, be aware of the I/O to the CFRM couple data set that might be generated.

Understanding IXLUSYNC Version Support

The IXLUSYNC macro supports versions in the range of 0 - 2.

- Keywords not specifically noted here are supported by version 0 and subsequent versions of the IXLUSYNC macro.
- The following keywords are supported by version 1 and subsequent versions of the IXLUSYNC macro.

PROXYRESPONSE	SUBJCONTOKEN

- The following keyword is supported by all versions starting with version 2 and higher of the IXLUSYNC macro.

COMPCODE

Specify the version of the parameter list that you want generated with the PLISTVER keyword. See “Specifying a Macro Version Number” on page 5 for considerations when specifying the version of the parameter list with PLISTVER.

Syntax Diagram

The syntax of the IXLUSYNC macro is as follows:

main diagram

```
▸▸──IXLUSYNC──␢──CONTOKEN=contoken──,USEREVENT=userevent──────────────────────────────▸
```

```
                    ┌─,USERSTATE=ALL_ZEROES─┐
►─,REQUEST=──┬─SET─┼───────────────────────┼─────────────┬─,RETCODE=retcode──,RSNCODE=rsncode──────►
             │      └─,USERSTATE=userstate──┘             │
             ├─CONFIRM─┤ parameters-2 ├──────────────────┤
             └─CONFIRMSET─┤ parameters-1 ├─┤ parameters-2 ├┘

   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼──────────────────────────────────────────────────►◄
   │                    ┌─,0D─────┐     │
   ├─,MF=(L─,mfctrl─────┼─────────┼─)───┤
   │                    └─,mfattr─┘     │
   │                    ┌─,COMPLETE─┐   │
   └─,MF=(E─,mfctrl─────┼───────────┼─)─┘
                        └─,COMPLETE─┘
```

parameters-1

```
                                  ┌─,USERSTATE=ALL_ZEROES─┐
►►──,NEXTUSEREVENT=nextuserevent──┼───────────────────────┼──────────────────────────►◄
                                  └─,USERSTATE=userstate──┘
```

parameters-2

```
   ┌─,COMPCODE=0─────────┐   ┌─,PROXYRESPONSE=NO──────────────────────────────────┐
►►─┼─────────────────────┼───┼────────────────────────────────────────────────────┼──────►◄
   └─,COMPCODE=compcode──┘   └─,PROXYRESPONSE=YES──,SUBJCONTOKEN=subjcontoken─────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined.

,COMPCODE=0
,COMPCODE=*compcode*

Use this input parameter to specify a user-defined completion code value. When the sync point is reached, the system presents the highest completion code that was set by any confirming connector to the event exit. The system provides a default value of zero if you do not specify a completion code value.

For connectors that fail or disconnect at a time that they owe a user sync point confirmation, the system confirms the sync point with a completion code of IXLUSYNCFAILEDUSERCOMPCODE (X'0000FFFF'). Thus, if a given user completion code is to take precedence over the completion code set by the system, the user completion code must be greater than X'0000FFFF'. Similarly, if the completion code set by the system is to take precedence over a user completion code, the user completion code must be less than IXLUSYNCFAILEDUSERCOMPCODE (X'0000FFFF').

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input field that contains the user-defined completion code value.

,CONTOKEN=*contoken*

Use this input parameter to specify the original connect token returned to the requestor from the IXLCONN macro. The IXLCONN macro allows the requestor to connect to the structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the connect token returned to the requestor when connecting to the structure with IXLCONN.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**

,MF=(E,*mfctrl*)
,MF=(E,*mfctrl*,COMPLETE)

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl

Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr

Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE

Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,NEXTUSEREVENT=*nextuserevent*

Use this input/output parameter to specify a value associated with the next user event to be set. The NEXTUSEREVENT value must be non-zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword input/output field to contain the value associated with the next user event.

,PROXYRESPONSE=NO
,PROXYRESPONSE=YES

Use this input parameter to indicate whether this response is being provided on behalf of a failing connector.

NO

Indicates that this is not a proxy response. The response is on behalf of the connector described by the CONTOKEN keyword.

YES

Indicates that this in a proxy response. The response is being provided on behalf of the connector described by the SUBJCONTOKEN keyword.

Note: To ensure that the PROXYRESPONSE feature is installed on the system on which you are running, issue IXCQUERY REQINFO=FEATURES. QUREQRFPROXYRESPONSE, returned from the IXCQUERY request, indicates whether the PROXYRESPONSE feature is installed.

,RETCODE=*retcode*
Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the return code.

,REQUEST=SET
,REQUEST=CONFIRM
,REQUEST=CONFIRMSET
Use this input parameter to identify the type of user sync point request. You can set a user event, confirm that processing is complete for a user event, or confirm that processing is complete and then set the next user event. Only one user event can be set at a time.

SET
Set the user event. When setting an event, a previous event, if there was one, must have been confirmed by all connectors and all connectors must have been told about the completion of the user sync point through their event exit. If the user event is successfully set, the User Sync Point event is presented to the event exit of each connector. The following information is presented to each connector in the event exit parameter list (IXLYEEPL):

- EEPLCOMPLETEDUSEREVENT — set to zero.
- EEPLNEXTUSEREVENT — set to the value specified by the USEREVENT keyword.
- EEPLCOMPLETEDUSERSTATE — set to zero.
- EEPLNEXTUSERSTATE — set to the value specified by the USERSTATE keyword or zero if USERSTATE is not specified.
- EEPLCOMPLETEDUSERCOMPCODE — set to zero.

CONFIRM
Confirm a user event specified by the USEREVENT keyword. When all connectors have confirmed the event and their confirmations have been received, the system presents the User Sync Point event to the event exit of each connector to indicate that a user sync point has been reached. The following information is presented to each connector in the event exit parameter list (IXLYEEPL):

- EEPLCOMPLETEDUSEREVENT — set to the value specified by the USEREVENT keyword.
- EEPLNEXTUSEREVENT — set to zero.
- EEPLCOMPLETEDUSERSTATE — set to the value specified when the event was set (with the USERSTATE keyword).
- EEPLNEXTUSERSTATE — set to zero.
- EEPLCOMPLETEDUSERCOMPCODE — set to the highest user completion code value that was specified by any confirming connector. Note that if a connector fails or disconnects while XES is expecting a user sync point confirmation from that connector, XES confirms the event for the failing connector with a completion code of IXLUSYNCFAILEDUSERCOMPCODE (X'0000FFFF').

CONFIRMSET
Confirm the user event specified by the USEREVENT keyword, and if this is the last confirmation for the event, set the next user event to the value specified by the NEXTUSEREVENT keyword. If this is not the last confirmation for the event, the “set” is ignored.

When all confirmations have been received, the system presents the User Sync Point event to the event exit of each connector indicating that a sync point has been reached. In the same parameter list, the next user event also is presented to each connector. The following information is presented to each connector in the event exit parameter list (IXLYEEPL):

- EEPLCOMPLETEDUSEREVENT — set to the value specified by the USEREVENT keyword.
- EEPLNEXTUSEREVENT — set to the value specified by the NEXTUSEREVENT keyword.
- EEPLCOMPLETEDUSERSTATE — set to the user state specified by the connector that set the original event.

- EEPLNEXTUSERSTATE — set to the value specified by the USERSTATE keyword or zero if USERSTATE is not specified.
- EEPLCOMPLETEDUSERCOMPCODE — set to the highest user completion code value that was specified by any confirming connector. Note that if a connector fails or disconnects while XES is expecting a user sync point confirmation from that connector, XES confirms the event for the failing connector with a completion code of IXLUSYNCFAILEDUSERCOMPCODE (X'0000FFFF').

,RSNCODE=*rsncode*
Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword to contain the reason code.

,SUBJCONTOKEN=*subjcontoken*
Use this input parameter to identify the connector for whom the response is being provided. The subject connector must be in the failing state.

Do not provide a proxy response on behalf of a failing connector until AFTER the EEPLDISCFAILCONNECTION event for the subject user has been presented to your event exit. You can use the value of the EEPLSUBJCONTOKEN field from that invocation of the event exit as input for this parameter.

Note that if the subject connector is not in the failing state, the IXLUSYNC request will fail with the IXLRSNCODESUBJCONNNOTFAILING reason code.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character input field that contains the connect token for which the response is being provided.

,USEREVENT=*userevent*
Use this input parameter to specify the user event associated with this request. The value of USEREVENT must be non-zero.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword input/output field that contains the value associated with the user event.

,USERSTATE=ALL_ZEROES
,USERSTATE=*userstate*
Use this input/output parameter to specify a user-defined value to be presented to the event exit. If this value is not provided, the field is set to the default of all zeroes. The user state presented to the event exit is that set by the connector that successfully set the user event.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 32-character input/output field that contains the user-defined value associated with the event that is to be presented to the event exit.

Return and Reason Codes

When the IXLUSYNC macro returns control to your program:
- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code, if applicable.

Macro IXLYCON provides equate symbols for the return and reason codes. The equate symbols associated with each hexadecimal return code are as follows:

0	IXLRETCODEOK
4	IXLRETCODEWARNING
8	IXLRETCODEPARMERROR
C	IXLRETCODEENVERROR
10	IXLRETCODECOMPERROR

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 82. Return and Reason Codes for the IXLUSYNC Macro

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
0	None.	**Meaning:** Processing completed successfully. **Action:** None.
4	xxxx0417	**Equate Symbol:** IXLRSNCODENOTLASTCONFIRMATION **Meaning:** Request to confirm the synchronization point has completed, but all connection confirmations have not yet been received. The next synchronization point has not yet been set. Applies only to REQUEST=CONFIRMSET **Action:** Request that the next event be set when all confirmations have been received for this event.
4	xxxx0420	**Equate Symbol:** IXLRSNCODEUSYNCEVENTSET **Meaning:** The user event specified has already been set by a peer connector. **Action:** Ensure that only one user-event is set at a time.
8	xxxx0801	**Equate Symbol:** IXLRSNCODEBADPARMLIST **Meaning:** The IXLUSYNC parameter list is not accessible. **Action:** Verify that: • The parameter list is uncorrupted. • The parameter list is addressable in the caller's primary address space. • If you are invoking IXLUSYNC in AR-mode and you specified the parameter list address using explicit register notation, the corresponding access register must be updated appropriately. • If you are invoking IXLUSYNC in AR-mode, SYSSTATE ASCENV=AR must be issued before the IXLUSYNC.
8	xxxx0802	**Equate Symbol:** IXLRSNCODEBADPARMLISTALET **Meaning:** The IXLUSYNC parameter list ALET is not valid. **Action:** Ensure that the ALET is zero or that the ALET represents a valid entry on the DU-AL.
8	xxxx0804	**Equate Symbol:** IXLRSNCODEBADVERSION# **Meaning:** There is an invalid version number in the IXLUSYNC parameter list. **Action:** • Verify that your program did not overlay the parameter list storage. • Verify that your program was assembled with the correct macro library for the release of MVS that your program is running on.
8	xxxx0806	**Equate Symbol:** IXLRSNCODESRBMODE **Meaning:** The requestor is in SRB mode. **Action:** Do not issue the IXLUSYNC macro when running in SRB mode.

Table 82. Return and Reason Codes for the IXLUSYNC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
8	xxxx0807	**Equate Symbol:** IXLRSNCODENOTENABLED **Meaning:** The requestor is not in an enabled state. **Action:** Issue the IXLUSYNC macro when running enabled for I/O and external interrupts.
8	xxxx0809	**Equate Symbol:** IXLRSNCODEPRIMARYNOTHOME **Meaning:** The primary address space does not equal the home address space. The request fails. **Action:** Make sure that the primary address space and the home address space are the same at the time of the IXLUSYNC invocation.
8	xxxx080A	**Equate Symbol:** IXLRSNCODEBADCONTOKEN **Meaning:** The requestor specified a CONTOKEN that is not valid. The CONTOKEN is invalid for one of the following reasons: disconnect has occurred, EOT of the connector's task, the input contoken is not the original contoken returned from IXLCONN, or the request was issued outside the connector's address space. **Action:** Verify that the CONTOKEN value specified was valid and for the correct structure.
8	xxxx0858	**Equate Symbol:** IXLRSNCODEBADUSEREVENT **Meaning:** The value specified for the USEREVENT or NEXTUSEREVENT keywords must be non-zero. **Action:** Correct the value specified for the USEREVENT or NEXTUSEREVENT keywords and resubmit the request.
8	xxxx0863	**Equate Symbol:** IXLRSNCODETASKTERM **Meaning:** The system rejects the request because the requesting task is going through termination. IXLUSYNC cannot be issued from a resource manager. **Action:** Examine your protocol to ensure that IXLUSYNC is not issued from a resource manager.
C	xxxx0C37	**Equate Symbol:** IXLRSNCODEUSEREVENTMISMATCH **Meaning:** Environment error. When confirming an event with REQUEST=CONFIRM or REQUEST=CONFIRMSET, the user event specified does not match the currently defined user event. **Action:** Ensure that the value of USEREVENT was specified correctly.
C	xxxx0C38	**Equate Symbol:** IXLRSNCODEUSERMISMATCH **Meaning:** Environmental error. The system did not expect a confirmation request from the responding connector. The connector probably has already confirmed the user event. **Action:** Check code to determine why the confirmation request was issued more than once.

Table 82. Return and Reason Codes for the IXLUSYNC Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Equate Symbol Meaning and Action
C	xxxx0C3F	**Equate Symbol:** IXLRSNCODECONNNOTDEFINED **Meaning:** Environmental error. The requesting connection is not defined. **Action:** Ensure that the IXLUSYNC request is issued by an active connector.
C	xxxx0C40	**Equate Symbol:** IXLRSNCODECONNNOTACTIVE **Meaning:** Environmental error. The requesting connection is not active. **Action:** Ensure that the IXLUSYNC request is issued by an active connector.
C	xxxx0C48	**Equate Symbol:** IXLRSNCODESUBJCONNNOTDEFINED **Meaning:** Environmental error. The connection identified by SUBJCONTOKEN is in the not defined state. **Action:** Verify that your parameter list has not been overlaid. Verify that SUBJCONTOKEN is from EEPLSUBJCONTOKEN from the EEPL passed to the event exit.
C	xxxx0C4B	**Equate Symbol:** IXLRSNCODEUSYNCEVENTNOTSET **Meaning:** Environmental error. When setting an event with REQUEST=SET, the system rejects the new user event because another event is still set. Either all confirmations have not been received for the current event or all connectors have not been notified of the previously completed event through the event exit. **Action:** Only one event can be set at a time.
C	xxxx0C4F	**Equate Symbol:** IXLRSNCODEUSYNCNOEVENTSET **Meaning:** Environmental error. A REQUEST=CONFIRM or REQUEST=CONFIRMSET is rejected because no user event is currently set. **Action:** Ensure that REQUEST=SET is issued before REQUEST=CONFIRM or REQUEST=CONFIRMSET.
C	xxxx0C6D	**Equate Symbol:** IXLRSNCODESUBJCONNNOTFAILING **Meaning:** Environmental error. An attempt to respond by proxy on behalf of the connector identified by the value provided for the SUBJCONTOKEN keyword failed because that connector is not in the failing state. **Action:** Ensure that you do not respond by proxy on behalf of a failing connector until after you have been presented with the EEPLDISCFAILCONNECTION event exit.
C	FFFFFFFF	**Equate Symbol:** IXLRSNCODENOTAVAILABLE **Meaning:** Environmental error. XES functions are not available. This can occur because the coupling facility hardware necessary to provide XES functions is not present. **Action:** Re-IPL the system, or follow your particular failure management protocol.

Table 82. Return and Reason Codes for the IXLUSYNC Macro (continued)

Hexadecimal Return Code	**Hexadecimal Reason Code**	**Equate Symbol Meaning and Action**
10	xxxx10xx	**Meaning:** XES processing has failed. **Action:** Save the reason code information and contact the IBM support center.

IXLVECTR — Check or Modify Local Cache Vector or List Notification Vector

Description

List structure users can use the IXLVECTR macro to check or modify their list notification vectors. Cache structure users can use the IXLVECTR macro to check or modify their local cache vectors.

Note: The following information assumes that you are familiar with either a list or cache services macro and that you are using either a list or cache structure.

If you are using a list structure:

Use the IXLVECTR macro to perform the following operations on your list notification vector.
- Modify the number of entries in your list notification vector.
- Test whether a list or an event queue is empty or nonempty.
- Test a range of vector entries to determine whether the monitored objects are empty or non-empty.

If you are using a cache structure:

Use the IXLVECTR macro to perform the following operations on your local cache vector:
- Modify the number of entries in your local cache vector.
- Determine the validity of data items in your local cache buffer.
- Test a range of vector entries to determine whether local cache vector entries are valid or invalid.

For More Information

The *z/OS MVS Programming: Sysplex Services Guide* contains a detailed description of the functions provided by the IXLVECTR macro as well as guidance for using the macro and examples of serialization protocols for use with the TESTLOCALCACHE request. Before using the IXLVECTR macro, be sure to read this information. The reference material provided here assumes you have done so.

Environment

The requirements for the caller are:

Minimum authorization:	For the MODIFYVECTORSIZE parameter, supervisor state. For all other parameters, problem state.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any HASN, any PASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or AR
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	If your program is disabled, you may hold the CPU lock. Otherwise, no locks may be held.
Control parameters:	Control parameters must be in the primary address space

Programming Requirements

1. If your program is in AR mode, issue SYSSTATE ASCENV=AR before you issue the IXLVECTR macro. ASCENV=AR causes the system to generate code appropriate for AR mode. For more information about the SYSSTATE macro, see *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

2. All virtual storage areas passed to the IXLVECTR macro must reside in your primary address space.
3. Include the IXLYCON mapping macro to generate the equate symbols for the return codes.
4. The correct use of the TESTLOCALCACHE request requires the implementation of a serialization protocol. Be sure to read the description of the necessary serialization protocol in *z/OS MVS Programming: Sysplex Services Guide*.

Restrictions

Registers 2-6 are used by the macro expansion. Therefore, registers 2-6 should not be used as base registers that will be used to resolve macro parameter addresses.

Input Register Information

Before issuing the IXLVECTR macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0-1	Unpredictable
2-13	Unchanged
14	Unpredictable
15	Return code

When control returns to the caller, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

1. You should specify VALIDATE=YES only with the first vector index that you hold serialization for. The remaining entries, under the same serialization, should be checked with VALIDATE=NO. Performance is significantly slower with VALIDATE=YES.
2. The vector indexes are local to the CPC you are running on and do not reside on the coupling facility.

Syntax Diagram

The syntax of the IXLVECTR macro is as follows:

main diagram

```
►►──IXLVECTR──␢──VECTORTOKEN=vectortoken──,WORKAREA=workarea──────────────────────────►

►─,REQUEST=─┬─MODIFYVECTORSIZE──┤ parameters-1 ├─┬──┬────────────────┬──────────────►◄
            ├─LTVECENTRIES──┤ parameters-3 ├─────┤  └─,RETCODE=retcode─┘
            ├─TESTLISTSTATE──┤ parameters-5 ├────┤
            └─TESTLOCALCACHE──┤ parameters-7 ├───┘
```

parameters-1

```
                                                  ┌─,BRANCHTABLE=NO──────────────────────┐
►►──,VECTORLEN=vectorlen──,ACTUALVECLEN=actualveclen─┼──────────────────────────────────────┼──────────►◄
                                                  └─,BRANCHTABLE=YES─┤ parameters-2 ├───┘
```

parameters-2

```
►►──,MODIFYDONE=modifydone──,LESSTHAN=lessthan──,NOSTORAGE=nostorage──,INVALIDTOKN=invalidtokn──────►

►─,INVALIDLEN=invalidlen──────────────────────────────────────────────────────────────────────────►◄
```

parameters-3

```
                                          ┌─,BRANCHTABLE=NO──────────────────────┐
►►──,VECTORINDEX=vectorindex──,BITSTRING=bitstring─┼──────────────────────────────────────┼──────────►◄
                                          └─,BRANCHTABLE=YES─┤ parameters-4 ├───┘
```

parameters-4

```
►►──,ALLEMPVAL=allempval──,SOMENEINV=someneinv──,INDXINVALID=indxinvalid──,INVALIDTOKN=invalidtokn────►◄
```

parameters-5

```
                             ┌─,BRANCHTABLE=NO──────────────────────┐
►►──,VECTORINDEX=vectorindex─┼──────────────────────────────────────┼──────────────────────►◄
                             └─,BRANCHTABLE=YES─┤ parameters-6 ├───┘
```

parameters-6

```
►►──,LSTEMPTY=lstempty──,LSTNONEMPTY=lstnonempty──,INDXINVALID=indxinvalid──,INVALIDTOKN=invalidtokn───►◄
```

parameters-7

```
    ┌─,VALIDATE=YES,VECTORINDEX=NO_VECTORINDEX──────────┐
►►──┼───────────────────────────────────────────────────┼──────────────────────────────►◄
    │                ┌─,VECTORINDEX=NO_VECTORINDEX─┐    │
    ├─,VALIDATE=YES──┼─────────────────────────────┼────┤
    │                └─,VECTORINDEX=vectorindex────┘    │
    └─,VALIDATE=NO───,VECTORINDEX=vectorindex───────────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined:

,ACTUALVECLEN=*actualveclen*
: Use this output parameter to specify a fullword field to receive the count of the new number of entries for the list notification vector or local cache vector after your MODIFYVECTORSIZE request has been processed. This field is valid only when a return code of X'4' is returned.

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field to contain the new vector length if it differs from the requested vector length.

,ALLEMPVAL=*allempval*
: Use this input parameter to specify the label to branch to if all monitored objects (comprised of lists and/or the user's event queue) in the range of vector entries are EMPTY (list notification vector), or if all vector entries are VALID (local cache vector).

  To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,BITSTRING=*bitstring*
: BITSTRING is a required output field that contains vector entry state information for the range of

vector entries specified. This field contains 32 bits. The first bit represents the first vector index entry specified on the VECTORINDEX parameter and continues up to a maximum of 32 vector index entries. The bits are interpreted as follows:

- 0 - the vector entry corresponding to this bit position indicates that the monitored list or event queue is NON-EMPTY (list notification vector) or that the local cache entry is INVALID (local cache vector).
- 1 - the vector entry corresponding to this bit position indicates that the monitored list or event queue is EMPTY (list notification vector) or that the local cache entry is VALID (local cache vector).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field.

,BRANCHTABLE=NO
,BRANCHTABLE=YES

Use BRANCHTABLE=NO to specify that IXLVECTR should not generate a branch table.

With the MODIFYVECTORSIZE request, use BRANCHTABLE=YES to specify that IXLVECTR should generate a branch table using the labels you specify for MODIFYDONE, LESSTHAN, NOSTORAGE, INVALIDTOKN, and INVALIDLEN.

With the LTVECENTRIES request, use BRANCHTABLE=YES to specify that IXLVECTR should generate a branch table using the labels you specify for ALLEMPVAL, SOMENEINV, INDXINVALID, and INVALIDTOKN.

With the TESTLISTSTATE request, use BRANCHTABLE=YES to specify that IXLVECTR should generate a branch table for you using the labels you specify for LSTEMPTY, LSTNONEMPTY, INDXINVALID, and INVALIDTOKN.

,INDXINVALID=*indxinvalid*

Use this input parameter to specify the label to branch to if the IXLVECTR service detects that the index value you specified is not valid.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,INVALIDLEN=*invalidlen*

Use this input parameter to specify the label to branch to if the value you specify using the VECTORLEN parameter is not valid.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,INVALIDTOKN=*invalidtokn*

Use this input parameter to specify the label to branch to if the vector token you specified was not valid.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,LESSTHAN=*lessthan*

Use this input parameter to specify the label to branch to if the IXLVECTR service cannot obtain sufficient storage to enlarge your list notification vector or local cache vector to the size you requested.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,LSTEMPTY=*lstempty*

Use this input parameter to specify the label to branch to if the IXLVECTR service finds the list or event queue of interest empty.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,LSTNONEMPTY=*lstnonempty*

Use this input parameter to specify the label to branch to if the IXLVECTR service finds the list or event queue of interest non-empty.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,MODIFYDONE=*modifydone*
Use this input parameter to specify the label to branch to if the IXLVECTR service is able to modify the list notification vector or local cache vector as requested.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,NOSTORAGE=*nostorage*
Use this input parameter to specify the label to branch to if the IXLVECTR service cannot obtain any storage to enlarge your list notification vector or local cache vector to the size you requested. The vector's size remains the same.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,REQUEST=MODIFYVECTORSIZE
,REQUEST=LTVECENTRIES
,REQUEST=TESTLISTSTATE
,REQUEST=TESTLOCALCACHE
Use REQUEST=MODIFYVECTORSIZE to modify the size of your list notification vector or local cache vector.

Use REQUEST=LTVECENTRIES to load and test a range of vector entries associated with a local vector for either a list or cache structure.

Use REQUEST=TESTLISTSTATE to test whether a list or an event queue you are monitoring is empty or non-empty.

Use REQUEST=TESTLOCALCACHE to test whether a data item in your local cache buffer is valid.

,RETCODE=*retcode*
Use this output parameter to specify a fullword field to contain the return code, which is also returned in register 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field.

,SOMENEINV=*someneinv*
Use this input parameter to specify the label to branch to if some monitored objects (comprised of lists and/or the user's event queue) in the range of vector entries are NON-EMPTY (list notification vector), or if some entries in the range of vector entries are INVALID (local cache vector).

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the label.

,VALIDATE=YES
,VALIDATE=NO
Use VALIDATE=YES with TESTLOCALCACHE as follows:

- When you omit VECTORINDEX, to request that the system validate connectivity to the coupling facility
- When you specify VECTORINDEX, to request that the system validate connectivity to the coupling facility and check the validity of a particular data item

Use VALIDATE=NO with TESTLOCALCACHE and VECTORINDEX to request that the system check the validity of a particular data item without validating connectivity to the coupling facility. You would choose this option only if a previous serialized request was done with VALIDATE=YES and the serialization has not yet been released.

,VECTORINDEX=*vectorindex*
With the LTVECENTRIES request, use this input parameter to specify the starting index of the range of vector entries that are to be loaded and tested. The index specified must be evenly divisible by 32. Thirty two consecutive vector entries will be loaded and tested. The vector index for a given vector size of N goes from zero to N-1.

With the TESTLISTSTATE request, use this input parameter to specify a fullword field that contains the vector index entry associated with the list or event queue of interest. For a vector with N entries, valid vector index values range from zero to N-1. The association between the particular index number and the list or event queue must already be established using the IXLLIST macro.

With the TESTLOCALCACHE request, use this input parameter to specify a fullword field that contains the vector index entry to be tested for validity. For a vector with N entries, valid vector index values range from zero to N-1. The association between the particular index number and the data item must already be established using the IXLCACHE macro.

For more information about the IXLLIST or IXLCACHE macros, see *z/OS MVS Programming: Sysplex Services Guide*.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field.

,VECTORLEN=*vectorlen*
Use this input parameter to specify a fullword field containing the new total number of entries in the list notification vector or local cache vector. The new number of entries can be greater than or less than the current number of entries but must be greater than zero and be a multiple of 32. If you specify a number that is not a multiple of 32, the system will round up the number to a multiple of 32.

The system will attempt to expand or contract your vector to the size you specify.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the requested vector length.

VECTORTOKEN=*vectortoken*
Use this input parameter to identify the list notification vector or local cache vector on which the specified request is to be performed.

The local vector was initially created by the IXLCONN service, if requested, and the vector token was returned in the connect answer area (IXLYCONA) in the field CONAVECTORTOKEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 12-character field that contains the vector token.

,WORKAREA=*workarea*
Use this input parameter to specify a 20-byte save area to be used by the IXLVECTR service.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 20-character field that begins on a word boundary.

ABEND Codes

Abend X'026' (See *z/OS MVS System Codes* for more information on this abend.

Return Codes

When the IXLVECTR macro returns control to your program, GPR 15 (and *retcode* if you coded RETCODE) contains a hexadecimal return code.

The IXLYCON macro provides equate symbols for the return and reason codes.

The following tables contain hexadecimal return codes, their associated equate symbols, and the meaning and suggested action for each return code issued by each type of IXLVECTR REQUEST.

Table 83. Return Codes for the IXLVECTR Macro with the MODIFYVECTORSIZE Parameter

Hexadecimal Return Code	Equate Symbol Meaning and Action
0	**Equate Symbol:** IXLRETCODEMODIFYDONE **Meaning:** The list notification vector or local cache vector was modified as requested. **Action:** None.

Table 83. Return Codes for the IXLVECTR Macro with the MODIFYVECTORSIZE Parameter (continued)

Hexadecimal Return Code	**Equate Symbol Meaning and Action**
4	**Equate Symbol:** IXLRETCODELESSTHAN **Meaning:** System error. The list notification vector or local cache vector is smaller than the size you requested because insufficient storage was available. The new number of vector entries is returned in the field specified by ACTUALVECLEN. **Action:** If you require a larger list notification vector, retry the request later when more storage might be available. If the problem persists, notify the system programmer so that the cause of the problem can be determined and corrected.
8	**Equate Symbol:** IXLRETCODENOSTORAGE **Meaning:** System error. Storage could not be obtained to increase the list notification vector or local cache vector size. The size remains unchanged. **Action:** Retry the request later when more storage might be available. If the problem persists, notify the system programmer so that the cause of the problem can be determined and corrected.
C	**Equate Symbol:** IXLRETCODEINVALIDTOKN **Meaning:** Program error. The vector token you specified is not valid. **Action:** Check your program for errors such as: • You specified the address of the vector token incorrectly. • The vector token has been overlaid or corrupted between the time you received it and the time you specified it on the IXLVECTR macro. • You disconnected from the cache structure (using the IXLDISC macro) before issuing the IXLVECTR macro. You must be a connected user to issue IXLVECTR. • The connector’s task failed and the vector was cleaned up. Correct the error, and rerun the program.
10	**Equate Symbol:** IXLRETCODEINVALIDLEN **Meaning:** Program error. The vector length you specified is not valid. **Action:** The vector length must be greater than, or equal to, 1. Correct the error, and rerun the program.

Table 84. Return Codes for the IXLVECTR Macro with the LTVECENTRIES Parameter

Hexadecimal Return Code	**Equate Symbol Meaning and Action**
0	**Equate Symbol:** IXLRETCODEALLEMPVAL **Meaning:** All monitored objects in the range of vector entries are empty (list notification vector), or all local cache entries in the range of vector entries are valid (local cache vector). **Action:** None.
4	**Equate Symbol:** IXLRETCODESOMENEINV **Meaning:** Some list or event queue in the range of vector entries is non-empty (list notification vector), or some local cache entry in the range of vector entries is invalid (local cache vector). **Action:** None.

Table 84. Return Codes for the IXLVECTR Macro with the LTVECENTRIES Parameter (continued)

Hexadecimal Return Code	Equate Symbol Meaning and Action
8	**Equate Symbol:** IXLRETCODEINDXINVALID **Meaning:** Program error. The vector index you specified is not valid. **Action:** Check your program for errors such as: • You specified a vector index that is not a multiple of 32. • You specified a vector index value less than zero. • You specified a vector index value greater than, or equal to, the number of vector entries. Vector index values for a vector with N entries range from zero to N-1. • You specified the address of the vector index value incorrectly. Correct the error, and rerun the program.
C	**Equate Symbol:** IXLRETCODEINVALIDTOKN **Meaning:** Program error. The vector token you specified is not valid. **Action:** Check your program for errors such as: • You specified the address of the vector token incorrectly. • The vector token has been overlaid or corrupted between the time you received it and the time you specified it on the IXLVECTR macro. • You disconnected from the list structure (using the IXLDISC macro) before issuing the IXLVECTR macro. You must be a connected user to issue IXLVECTR. • The connector's task failed and the vector was cleaned up. Correct the error, and rerun the program.

Table 85. Return Codes for the IXLVECTR Macro with the TESTLISTSTATE Parameter

Hexadecimal Return Code	Equate Symbol Meaning and Action
0	**Equate Symbol:** IXLRETCODELSTEMPTY **Meaning:** The list or event queue is empty. **Action:** None.
4	**Equate Symbol:** IXLRETCODELSTNONEMPTY **Meaning:** The list or event queue is not empty. **Action:** None.
8	**Equate Symbol:** IXLRETCODEINDXINVALID **Meaning:** Program error. The vector index you specified is not valid. **Action:** Check your program for errors such as: • You specified a vector index value greater than, or equal to, the number of vector entries. Vector index values for a vector with N entries range from zero to N-1. • You specified the address of the vector index value incorrectly. • Another unit of work modified the vector size while this TESTLISTSTATE was being processed. Correct the error, and rerun the program.

Table 85. Return Codes for the IXLVECTR Macro with the TESTLISTSTATE Parameter (continued)

Hexadecimal Return Code	**Equate Symbol Meaning and Action**
C	**Equate Symbol:** IXLRETCODEINVALIDTOKN **Meaning:** Program error. The vector token you specified is not valid. **Action:** Check your program for errors such as: • You specified the address of the vector token incorrectly. • The vector token has been overlaid or corrupted between the time you received it and the time you specified it on the IXLVECTR macro. • You disconnected from the list structure (using the IXLDISC macro) before issuing the IXLVECTR macro. You must be a connected user to issue IXLVECTR. • The connector's task failed and the vector was cleaned up. Correct the error, and rerun the program.

Table 86. Return Codes for the IXLVECTR Macro with the TESTLOCALCACHE Parameter

Hexadecimal Return Code	**Equate Symbol Meaning and Action**
0	**If you specified the VECTORINDEX parameter:** **Equate Symbol:** IXLRETCODEBUFVALID **Meaning:** The data item in the local cache buffer is valid. **Action:** None. **If you omitted the VECTORINDEX parameter:** **Equate Symbol:** IXLRETCODECONNECTED **Meaning:** There has been no interruption of connectivity to the coupling facility. Cross invalidation is reflected for data items in the cache structure. **Action:** None.
4	**If you specified the VECTORINDEX parameter:** **Equate Symbol:** IXLRETCODEBUFNOTVALID **Meaning:** The data item in the local cache buffer is not valid. **Action:** Obtain new copy if the data item. **If you omitted the VECTORINDEX parameter:** **Equate Symbol:** IXLRETCODENOTCONNECTED **Meaning:** All vector index entries are invalid. **Action:** Obtain new copies of data items.
8	**Equate Symbol:** IXLRETCODEINDXINVALID **Meaning:** Program error. TESTLOCALCACHE was specified with a VECTORINDEX value that is not valid. **Action:** Check your program for errors such as: • You specified a vector index value greater than, or equal to, the number of vector entries. • You specified the address of the vector index value incorrectly. • Another unit of work modified the vector size while this TESTLOCALCACHE was being processed. Correct the error, and rerun the program.

Table 86. Return Codes for the IXLVECTR Macro with the TESTLOCALCACHE Parameter (continued)

Hexadecimal Return Code	Equate Symbol Meaning and Action
C	**Equate Symbol:** IXLRETCODEINVALIDTOKN **Meaning:** Program error. The vector token you specified is not valid. **Action:** Check your program for errors such as: • You specified the address of the vector token incorrectly. • You disconnected from the cache structure (using the IXLDISC macro) before issuing the IXLVECTR macro. You must be a connected user to issue IXLVECTR. • The vector token has been overlaid or corrupted between the time you received it and the time you specified it on the IXLVECTR macro. • The connector's task failed and the vector was cleaned up. Correct the error, and rerun the program.

IXLZSTR — Coupling Facility Structure Data Access Service

Description

The IXLZSTR macro allows you to request coupling facility structure data from a dump containing that information. The macro returns the data requested in an answer area that you provide. You can use IXLZSTR only in an IPCS environment.

The TYPE parameter on the IXLZSTR macro determines what type of data will be returned to the caller:
- TYPE=STRUCTURE returns structure information.
- TYPE=CLASS returns class information for a requested cache structure.
- TYPE=LISTNUM returns list number information for a requested list structure.
- TYPE=LOCKENTRIES returns lock table information for a requested list structure.
- TYPE=USERCNTLS returns user control information for a requested structure.
- TYPE=ENTRY returns entry information for a requested entry in a requested structure.
- TYPE=EMCONTROLS returns event monitor control information for a requested list structure. The requested list structure must be a keyed list structure allocated in a coupling facility with CFLEVEL=4 or higher.
- TYPE=EVENTQS returns event queue control information for a requested connection ID. The requested connection ID must be connected to a keyed list structure in a coupling facility with CFLEVEL=4 or higher.

To use the macro, follow these guidelines:
- Ensure that you include the proper structure attributes on the applicable structure type. For example, you cannot request a range of list numbers for a cache structure or a castout class for a list structure. If you specify structure attributes that do not apply, IXLZSTR fails the request and returns a return code of 8 and a reason code of X'14'.

  Note: To determine the structure type of a given structure in the dump, issue the IXLZSTR macro with TYPE=STRUCTURE,STRLEVEL=SUMMARY. IXLZSTR returns a list of all structures requested to be dumped and their structure type in the answer area. Using the StrBStrSummary mapping in IXLZSTRB, you can find the structure name and check its corresponding type.

  To determine the correct dump identifier for a given structure, issue the same macro as above. IXLZSTR returns the structure dump ID for each structure in the list of structures requested to be dumped, as well as the name of the coupling facility in which the structure was allocated.
- On requests that specify a range of values, ensure that the starting range value is not greater than the ending range value. If the value is greater, IXLZSTR fails the request and returns a return code of 8 and a reason code of X'18'.
- Use the RESTOKEN keyword to retrieve information when IXLZSTR cannot return all information at once in the answer area. When this volume of information exists, you must issue the IXLZSTR macro more than once, each time specifying the RESTOKEN as input.

Determining an Answer Area Size

The IXLZSTR macro puts the requested information in an answer area that you provide. The answer area contains a header (STRBHEADER) that describes the remainder of the information retrieved, followed by one or more entries for the requested information.

The amount of storage that you can specify for the answer area depends on the type of request. The minimum answer area size is 4096 bytes (4K). If you supply less than this amount, IXLZSTR fails the request and returns a return code of 8 and a reason code of X'10'.

A minimum storage allocation of 8192 bytes (8K) is required for a TYPE=STRUCTURE STRLEVEL=DETAIL request and for any request that specifies the ENTRYDATA keyword. If you provide less than 8K of storage for the answer area, IXLZSTR fails the request and returns a return code of 8 and a reason code of X'10'.

You might not be able to provide an answer area large enough to contain all the information returned by IXLZSTR for a single request. To allow for this situation, IXLZSTR provides the RESTOKEN parameter. You specify this token, initialized to binary zeros, on each invocation of IXLZSTR. If all the information can be returned in the answer area, IXLZSTR sets a return code of 0 and a reason code of 0. However, if all the information cannot be returned at once, IXLZSTR sets a return code of 4 and a reason code of 4 to indicate that there is more data to be retrieved from the dump data set. To retrieve the remainder of the data, you must invoke the IXLZSTR macro again with the same keywords and with RESTOKEN as input. (Do not reset RESTOKEN to binary zeros until all until all data for this request is retrieved.) When all data has been retrieved, IXLZSTR sets a return code of 0 and a reason code of 0 to indicate successful completion. IXLZSTR also sets the RESTOKEN to binary zeros.

Determining the Size of Each Entry Returned

Each time you retrieve information into your answer area, the header record (STRBHEADER) specifies the length of the table entry (STRBTABLEENTRYLEN). Each table entry also contains a header (STRBENTRY), which contains additional length specifications for the data.

– STRBENTRYADJLEN — length of the adjunct data
– STRBENTRYEDATALEN — length of the entry data
– STRBENTRYCNTLLEN — length of the entry control information

By calculating the sum of these four lengths, you can determine the length of the table entry and calculate the location of the next entry in the answer area.

When the information requested cannot all fit in one answer area, you must again calculate the length for the entry. Additional entries that can fit in the answer area after the first is completed, are preceded by the STRBENTRY header, which contains the appropriate lengths for the data entry that follows.

Environment

Minimum authorization:	One of the following: 1. Problem state 2. PKM allowing key 8
Dispatchable unit mode:	Task
Amode:	31-bit
ASC mode:	Primary
Serialization:	**Enabled**
Control parameters:	Control parameters must be in the primary address space

Programming Requirements

The IXLZSTRB macro provides the format of the area that ANSAREA points to. If you intend to use that area, include that macro in your program. IXLZSTRB maps each of the following types of data to be returned to the caller:

- STRBHEADER maps the header section of the answer area.
- STRBSTRSUMMARY maps summary information about the requested structure.
- STRBSTRDETAIL maps detail information about a requested structure.
- STRBSUMMARY maps summary information for a specified CLASS, LISTNUM, or event monitor controls request for a particular structure.
- STRBDETAIL maps detail information for a specified CLASS, LISTNUM, or event queue controls request for a particular structure.
- STRBENTRY maps entry information for a specified CLASS or LISTNUM request for a particular structure.
- STRBEMCDETAIL maps EMC information for event monitor controls associated with a specified LISTNUM and maps EMC information for event queue controls associated with a specified CONID.

Depending on the request type, you might need additional macros to map the information returned by IXLZSTR.

Table 87. Answer Area Macros for IXLZSTR

Macro Name	Descriptive Name	Macro Use
IHAARB	Associated Request Block Mapping	Provides a map of the list of all the ranges of objects that were validly requested to be dumped.
IXLYDCAC	Dumping Coupling Facility Cache Structure Controls Mapping	Provides a map of the dumping cache structure controls.
IXLYDCCC	Dumping Cast-out Class Controls Mapping	Provides a map of the dumping castout class controls.
IXLYDDIB	Dumping Directory Information Block Mappings	Provides mappings for: • Lock table entry (LTE), which contains the lock table entry information associated with a structure. • List-entry control block (LECB), which contains the entry controls associated with a list structure. • Directory information block (DIFB), which contains the element controls associated with a cache structure. • List-user control block (LUCB), which contains the list user controls. • Local-cache control block (LCCB), which contains the local cache controls. • Event monitor controls block (EMC), which contains the event monitor controls associated with a list structure.
IXLYDEQC	Dumping Event Queue Controls Mapping	Provides a map of the dumping event queue controls.
IXLYDLC	Dumping List Controls Mapping	Provides a map of the dumping list header controls and the list monitor table entries found in the list controls.
IXLYDLCC	Dumping Local Cache Controls Mapping	Provides a map of the dumping local cache controls.
IXLYDLIC	Dumping Coupling Facility List Structure Controls Mapping	Provides a map of the dumping list structure controls.
IXLYDLUC	Dumping List User Controls Mapping	Provides a map of the dumping list user controls.
IXLYDSCC	Dumping Storage Class Controls Mapping	Provides a map of the dumping storage class controls.
IXLYSTRC	Coupling Facility Structure Data Access Service Dump Reason Code Constants	Provides the constants for interpreting the dump reason code.
IXLZSTRB	Coupling Facility Structure Data Access Service Answer Area Mappings	Maps the answer area data that was requested and provides constants to interpret any return and reason codes issued.

Restrictions

You can invoke IXLZSTR only in an IPCS environment. Include the BLSABDPL mapping macro.

z/OS MVS IPCS User's Guide provides general information about an IPCS environment.

Input Register Information

Before issuing the IXLZSTR macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller of the IXLZSTR macro, the general purpose registers (GPRs) contain:

Register	Contents
0	Reason code if GPR15 return code is non-zero
1	Used as work register by the system
2 - 13	Unchanged
14	Used as work register by the system
15	Return code

For registers that the system changes, a caller who depends on these registers containing the same value before and after issuing the macro must save these registers before issuing the macro and restore them after the system returns control.

Performance Implications

None.

Syntax Diagram

The syntax of the IXLZSTR macro is as follows:

main diagram

```
►►──IXLZSTR──b──ANSAREA=ansarea──,ANSLEN=anslen──,RESTOKEN=restoken──,ABDPLPTR=abdplptr──────────────►


►─,TYPE=──┬─STRUCTURE──┤ parameters-1 ├──┬──┬─────────────────┬──┬─────────────────┬──────────────►
          ├─CLASS──┤ parameters-2 ├──────┤  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
          ├─LISTNUM──┤ parameters-4 ├────┤
          ├─LOCKENTRIES──┤ parameters-6 ├┤
          ├─USERCNTLS──┤ parameters-7 ├──┤
          ├─ENTRY──┤ parameters-8 ├──────┤
          ├─EMCONTROLS──┤ parameters-9 ├─┤
          └─EVENTQS──┤ parameters-10 ├───┘


   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼───────────────────────────────────────────────────────►◄
   │                    ┌─,0D──────┐    │
   ├─,MF=(L─,mfctrl─────┼──────────┼─)──┤
   │                    └─,mfattr──┘    │
   │                    ┌─,COMPLETE─┐   │
   └─,MF=(E─,mfctrl─────┼───────────┼─)─┘
                        └─,COMPLETE─┘
```

parameters-1

```
►►──,──STRLEVEL──=──┬─SUMMARY──────────────────────────────────────┬────────────────────────────►◄
                    └─DETAIL─,STRNAME=strname─,STRDUMPID=strdumpid─┘
```

parameters-2

```
►►──,STRNAME=strname──,STRDUMPID=strdumpid──,CLASSTYPE=─┬─CASTOUT─┬──────────────────────────────────────►
                                                        └─STORAGE─┘

                          ┌─,STARTVAL=ALL──────────────────────────────┐
►─,CLASSLEVEL=─┬─SUMMARY──┴────────────────────────────────────────────┴─┬─────────────────────────────►◄
               │          └─,STARTVAL=startval─┬─,ENDVAL=STARTVAL─┬─┘   │
               │                               └─,ENDVAL=endval───┘     │
               │        ┌─,STARTVAL=ALL──────────────────────────────┐  │
               ├─DETAIL─┴────────────────────────────────────────────┴──┤
               │        └─,STARTVAL=startval─┬─,ENDVAL=STARTVAL─┬─┘    │
               │                             └─,ENDVAL=endval───┘      │
               └─ENTRY──┤ parameters-3 ├────────────────────────────────┘
```

parameters-3

```
                       ┌─,STARTPOS=ALL──────────────────────────────┐  ┌─,ORDER=HEAD─┐
►►──,CLASSVAL=classval─┴────────────────────────────────────────────┴──┴─────────────┴──────────────►
                       └─,STARTPOS=startpos─┬─,ENDPOS=STARTPOS─┬─┘     └─,ORDER=TAIL─┘
                                            └─,ENDPOS=endpos───┘

  ┌─,ADJUNCT=NO──┐  ┌─,ENTRYDATA=NO──┐
►─┴──────────────┴──┴────────────────┴──────────────────────────────────────────────────────────────►◄
  └─,ADJUNCT=YES─┘  └─,ENTRYDATA=YES─┘
```

parameters-4

```
►►──,STRNAME=strname──,STRDUMPID=strdumpid──────────────────────────────────────────────────────────►

                            ┌─,STARTVAL=ALL──────────────────────────────┐
►─,LISTNUMLEVEL=─┬─SUMMARY──┴────────────────────────────────────────────┴─┬───────────────────────►◄
                 │          └─,STARTVAL=startval─┬─,ENDVAL=STARTVAL─┬─┘   │
                 │                               └─,ENDVAL=endval───┘     │
                 │        ┌─,STARTVAL=ALL──────────────────────────────┐  │
                 ├─DETAIL─┴────────────────────────────────────────────┴──┤
                 │        └─,STARTVAL=startval─┬─,ENDVAL=STARTVAL─┬─┘    │
                 │                             └─,ENDVAL=endval───┘      │
                 └─ENTRY──┤ parameters-5 ├────────────────────────────────┘
```

parameters-5

```
                        ┌─,ENTRYKEY=NO_KEY──┐  ┌─,STARTPOS=ALL──────────────────────────────────┐
►►─,LISTNUMVAL=listnumval─┼───────────────────┼──┼────────────────────────────────────────────────┼──────►
                        └─,ENTRYKEY=entrykey─┘  │                         ┌─,ENDPOS=STARTPOS─┐ │
                                                 └─,STARTPOS=startpos─────┼──────────────────┼─┘
                                                                          └─,ENDPOS=endpos───┘

   ┌─,ORDER=HEAD─┐  ┌─,ADJUNCT=NO──┐  ┌─,ENTRYDATA=NO──┐
►──┼─────────────┼──┼──────────────┼──┼────────────────┼──────────────────────────────────────────────►◄
   └─,ORDER=TAIL─┘  └─,ADJUNCT=YES─┘  └─,ENTRYDATA=YES─┘
```

parameters-6

```
                                                   ┌─,STARTVAL=ALL──────────────────────────────┐
►►─,STRNAME=strname─,STRDUMPID=strdumpid───────────┼────────────────────────────────────────────┼──────►◄
                                                   │                      ┌─,ENDVAL=STARTVAL─┐ │
                                                   └─,STARTVAL=startval───┼──────────────────┼─┘
                                                                          └─,ENDVAL=endval───┘
```

parameters-7

```
                                                   ┌─,STARTVAL=ALL──────────────────────────────┐
►►─,STRNAME=strname─,STRDUMPID=strdumpid───────────┼────────────────────────────────────────────┼──────►◄
                                                   │                      ┌─,ENDVAL=STARTVAL─┐ │
                                                   └─,STARTVAL=startval───┼──────────────────┼─┘
                                                                          └─,ENDVAL=endval───┘
```

parameters-8

```
                                                         ┌─,ADJUNCT=NO──┐  ┌─,ENTRYDATA=NO──┐
►►─,STRNAME=strname─,STRDUMPID=strdumpid───┤ , ├─────────┼──────────────┼──┼────────────────┼──────►◄
                                         └─┤ , ├─┘       └─,ADJUNCT=YES─┘  └─,ENTRYDATA=YES─┘

├──ENTRYNAME=entryname──────────────────────────────────────────────────────────────────────┤

├──ENTRYID=entryid──────────────────────────────────────────────────────────────────────────┤
```

parameters-9

```
►►─,STRNAME=strname─,STRDUMPID=strdumpid──────────────────────────────────────────────────────────►

                          ┌─,STARTVAL=ALL──────────────────────────────┐
►─,EMCLEVEL=──┬─SUMMARY───┼────────────────────────────────────────────┼──┬────────────────────────►◄
              │           │                      ┌─,ENDVAL=STARTVAL─┐ │  │
              │           └─,STARTVAL=startval───┼──────────────────┼─┘  │
              │                                  └─,ENDVAL=endval───┘    │
              │                                  ┌─,ENTRYKEY=NO_KEY──┐   │
              └─EMC─,LISTNUMVAL=listnumval───────┼───────────────────┼───┘
                                                 └─,ENTRYKEY=entrykey─┘
```

parameters-10

```
►►─,STRNAME=strname─,STRDUMPID=strdumpid─────────────────────────────────────────►


                         ┌─,STARTVAL=ALL─────────────────────┐
►─,EQLEVEL=─┬─DETAIL─┼──────────────────────────────────────┼─┬─────────────────►◄
            │         │                    ┌─,ENDVAL=STARTVAL─┐ │ │
            │         └─,STARTVAL=startval─┤                  ├─┘ │
            │                              └─,ENDVAL=endval───┘   │
            └─EMC─,CONIDVAL=conidval──────────────────────────────┘
```

Parameter Descriptions

The parameter descriptions are listed in alphabetical order. Default values are underlined.

,ABDPLPTR=*abdplptr*

Use this input parameter to specify the address of the ABDUMP parameter list (ABDPL) that the system currently is using.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the address of the ABDPL.

,ADJUNCT=NO
,ADJUNCT=YES

Use this input parameter to specify whether the adjunct data associated with the requested entries should be returned with the entries. The STRBENTRY mapping for each entry points to the adjunct data and also provides the length of the adjunct data.

NO
Do not return adjunct data.

YES
Do return adjunct data.

ANSAREA=*ansarea*

Use this output parameter to identify the answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the answer area.

,ANSLEN=*anslen*

Use this input parameter to specify the length of the answer area.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the length of the answer area.

,CLASSLEVEL=SUMMARY
,CLASSLEVEL=DETAIL
,CLASSLEVEL=ENTRY

Use this input parameter to specify the level of the CLASS type request.

SUMMARY
Information about a requested class, a range of classes, or all classes (castout or storage) in the dump. The STRBSUMMARY mapping of IXLZSTRB maps the entries in the answer area.

DETAIL
Detailed information about a requested class, a range of classes, or all classes (castout or storage) in the dump. The STRBDETAIL mapping of IXLZSTRB maps the entries in the answer area. The entry also contains a pointer to the class controls information. For storage class information, the DSCC mapping in IXLYDSCC maps the storage class controls. For castout class information, the DCCC mapping in IXLYDCCC maps the castout class controls.

ENTRY
An entry at a requested position, a group of entries at a requested range of positions, or the entries at all entry positions that were dumped in a requested class. The STRBENTRY mapping of IXLZSTRB maps the entries in the answer area. The entry also contains a pointer to the entry control information. The DDIC mapping in IXLYDDIB maps the entry controls.

,CLASSTYPE=CASTOUT
,CLASSTYPE=STORAGE
Use this input parameter to specify the type of class that should be retrieved.

CASTOUT
Castout classes

STORAGE
Storage classes

,CLASSVAL=*classval*
Use this input parameter to identify the class for which data entries will be retrieved.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the class for which data entries will be retrieved.

,CONIDVAL=*conidval*
Use this input parameter to specify the connection ID for which the system will retrieve event monitor control entries on the event queue.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a one-byte field that contains the connection ID for which event monitor control information is being requested.

,EMCLEVEL=SUMMARY
,EMCLEVEL=EMC
Use this input parameter to specify the level of the EMCONTROLS type request.

SUMMARY
The list of event monitor controls in the list structure that was requested to be dumped. The system returns information for a list number, a range of list numbers, or all list numbers in the dump. The STRBSUMMARY mapping of the IXLZSTRB macro maps the entries in the answer area.

EMC
Event monitor controls that were dumped for the requested list structure. The STRBEMCDETAIL mapping of the IXLZSTRB macro maps the entries in the answer area. The STRBEMCDETAIL contains a pointer to the event monitor controls in the answer area. The DEMC mapping of the IXLYDDIB macro maps the event monitor controls entry.

,ENDPOS=STARTPOS
,ENDPOS=*endpos*
Use this input parameter to specify the end of the requested entry position range. If you do not specify this keyword, the ending range position will be equal to the starting range position specified on the STARTPOS keyword.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the end of the requested entry position range. ENDPOS is an optional keyword. The default is STARTPOS.

,ENDVAL=STARTVAL
,ENDVAL=*endval*
Use this input parameter to specify the end of the requested class range. If you do not specify this keyword, the ending range value will be equal to the starting range value specified on the STARTVAL keyword.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value indicating the end of the requested class range.

,ENTRYDATA=NO
,ENTRYDATA=YES

Use this input parameter to specify whether the entry data associated with the requested entries should be returned with the entries. The STRBENTRY mapping for each entry points to the entry data and also provides the total length of the entry data. If the buffer fills up before all entry data is returned, the STRBENTRY mapping provides the length of the entry data that has been returned.

NO

Do not return entry data.

YES

Do return entry data.

,ENTRYID=*entryid*

Use this input parameter to specify the list entry identifier of the list entry to be retrieved. This keyword should be used only if the requested structure is a list structure.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 12-character field that contains the list entry identifier of the list entry to be retrieved.

,ENTRYKEY=*entrykey*

Use this input parameter to identify the key to be used for retrieving list entries. If this keyword is specified, only list entries with the requested key will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the key to be used for retrieving list entries.

,ENTRYNAME=*entryname*

Use this input parameter to specify the name of the data entry or list entry to be retrieved.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 16-character field that contains the name of the data entry or list entry to be retrieved.

,EQLEVEL=DETAIL
,EQLEVEL=EMC

Use this input parameter to specify the level of the EVENTQS type request.

DETAIL

Detailed information about a requested connection ID, a range of connection IDs, or all connection IDs in the dump. The STRBDETAIL mapping of the IXZLSTRB macro maps the entries in the answer area. In the STRBDETAIL entry, there is a pointer to the event queue controls in the answer area. The DEQC mapping of the IXLYDEQC macro maps the event queue controls. The STRBDETAIL entry also contains the length of the event queue controls.

EMC

Event monitor controls entries associated with the event queue of the requested connection ID. The STRBEMCDETAIL mapping of the IXLZSTRB macro maps the entries in the answer area. In the STRBEMCDETAIL entry, there is a pointer to the event monitor controls in the answer area. The DEMC mapping of the IXLYDDIB macro maps the event monitor controls.

,LISTNUMLEVEL=SUMMARY
,LISTNUMLEVEL=DETAIL
,LISTNUMLEVEL=ENTRY

Use this input parameter to specify the level of the LISTNUM request. You can request summary information about all list numbers, detail information about specific list numbers, or information about entry positions.

SUMMARY

Information about the list numbers in the requested structure that was dumped. You can specify a specific list number, a range of list numbers, or all list numbers in the dump. The STRBSUMMARY mapping of IXLZSTRB maps the entries in the answer area.

DETAIL
Detailed information about a specific list number, a range of list numbers, or all list numbers in the dump. The STRBDETAIL mapping of IXLZSTRB maps the entries in the answer area. The STRBDETAIL mapping includes a pointer to the list controls in the answer area and the length of the list controls. The DLC mapping of IXLYDLC maps the list controls.

ENTRY
Information for a particular list number about an entry at a specific position, a group of entries at a range of positions, or the entries at all positions in the dump. The STRBENTRY mapping of IXLZSTRB maps the entries in the answer area. The STRBENTRY also contains a pointer to the entry controls in the answer area. The DDIL mapping of IXLYDDIB maps the entry controls.

,LISTNUMVAL=*listnumval*
Use this input parameter to specify the list number for which list entries are to be retrieved.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the list number for which entries are to be retrieved.

,MF=S
,MF=(L,*mfctrl***)**
,MF=(L,*mfctrl,mfattr***)**
,MF=(L,*mfctrl***,0D)**
,MF=(E,*mfctrl***)**
,MF=(E,*mfctrl***,COMPLETE)**
Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,mfctrl
Use this output parameter to specify a storage area to contain the parameters.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the parameter list.

,mfattr
Use this input parameter to specify the name of a 1- to 60-character string that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Use this input parameter to require that the system check for required parameters and supply defaults for omitted optional parameters.

Note: In the macro expansion you might see some defaults for optional parameters that are not documented here. The ones that are not documented do not have any effect on the macro. For example, if SMILE=*var* were an optional parameter and the default is SMILE=NO_SMILE then it would not be documented. However, if the default was SMILE=:-), then it would be documented because a value would be the default.

,ORDER=HEAD

,ORDER=TAIL

Use this input parameter to specify the order in which entries for the requested class are to be returned. For a storage class, the head of the queue represents the least recently referenced entry and the tail of the queue represents the most recently referenced entry. For a castout class, the head of the queue represents the most recently changed entry and the tail of the queue represents the least recently changed entry.

HEAD

Entries are in head-to-tail order. If you also specify the keywords STARTPOS/ENDPOS, the system returns the entries in head-to-tail order.

TAIL

Entries are in tail-to-head order. If you also specify the keywords STARTPOS/ENDPOS, the system returns the entries in tail-to-head order.

,RETCODE=*retcode*

Use this output parameter to specify the location in which the system is to copy the return code from GPR 15.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the return code.

,RESTOKEN=*restoken*

Use this input parameter to specify a 64-byte token that must be passed on all IXLZSTR requests. The token allows a request to be continued across multiple calls if all the requested data could not be returned in the answer area. Before your first request to access the data, you must initialize the field to binary zeros. On every request, the IXLZSTR service initializes RESTOKEN with information required on subsequent IXLZSTR requests. Therefore, the user who is requesting the data must not modify the contents of RESTOKEN.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the 64-character field that must be passed on all IXLZSTR requests.

,RSNCODE=*rsncode*

Use this output parameter to specify the location in which the system is to copy the reason code from GPR 0.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the reason code.

,STARTPOS=ALL
,STARTPOS=*startpos*

Use this input parameter to specify the start of the requested entry position range. If you do not specify this keyword, all of the entries in the dump for the requested class will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a fullword field that contains the start of the requested entry position range.

,STARTVAL=ALL
,STARTVAL=*startval*

Use this input parameter to specify the start of the requested class range. If you do not specify this keyword, all of the classes requested to be dumped will be returned.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of the fullword field that contains the value indicating the start of the requested class range.

,STRDUMPID=*strdumpid*

Use this input parameter to specify the structure dump ID of structure being requested.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a halfword field that contains the structure dump ID.

,STRLEVEL=SUMMARY

,STRLEVEL=DETAIL
Use this input parameter to specify the level of the STRUCTURE request.

SUMMARY
A list of all the structures and their types that are in the dump. STRBSTRSUMMARY maps the entries in the answer area.

DETAIL
Detailed information about a requested structure. STRBSTRDETAIL maps the entry in the answer area. The entry also contains a pointer to the structure control information. For a cache structure, the DCAC mapping in IXLYDCAC maps the structure controls. For a list structure, the DLIC mapping in IXLYDLIC maps the structure controls.

,STRNAME=*strname*
Use this input parameter to specify the name of the structure for which information is being requested.

To Code: Specify the RS-type name or address (using a register from 2 to 12) of a 16-character field that contains the name of the structure for which information is being requested.

,TYPE=STRUCTURE
,TYPE=CLASS
,TYPE=LISTNUM
,TYPE=LOCKINDEX
,TYPE=USER
,TYPE=ENTRY
,TYPE=EMCONTROLS
,TYPE=EVENTQS
Use this input parameter to specify the type of information to be retrieved by the IXLZSTR macro from the dump data set.

STRUCTURE
Structure information.

CLASS
Castout class or storage class information. Use CLASS only for a cache structure.

LISTNUM
List number information. Use LISTNUM only for a list structure.

LOCKENTRIES
Detailed information about a requested lock entry, a range of lock entries, or all entries in the lock table for a requested list structure. IXLZSTR returns only the non-zero lock table entries. Use LOCKENTRIES only for a list structure. The DLTE mapping in IXLYDDIB maps the entries in the answer area.

USERCNTLS
Detailed information about a requested connection ID, a range of connection IDs, or all connection IDs for a requested structure. For a cache structure, the DLCCB mapping in IXLYDDIB maps the entries in the answer area. For a list structure, the DLUCB mapping in IXLYDDIB maps the entries in the answer area.

ENTRY
Detailed information about a requested entry name or entry ID in a requested structure. The STRBENTRY mapping in IXLZSTRB maps the entry in the answer area. The entry also contains a pointer to the entry controls. For a cache structure, the DDIC mapping in IXLYDDIB maps the entry controls. For a list structure, the DDIL mapping in IXLYDDIB maps the entry controls.

EMCONTROLS
Event monitor control information. Use EMCONTROLS only for a keyed list structure allocated in a coupling facility with CFLEVEL=4 or higher. The DEMC mapping in IXLYDDIB maps the event montitor controls.

EVENTQS
Event queue control informaton. Use EVENTQS only for a keyed list structure allocated in a coupling facility with CFLEVEL=4 or higher. IXLYDEQC maps the event queue controls.

ABEND Codes

None.

Return and Reason Codes

When the system returns control to the caller, GPR 15 (and *retcode*, if you coded RETCODE) contains the return code and GPR 0 (and *rsncode*, if you coded RSNCODE) contains the reason code.

Table 88. Return and Reason Codes for the IXLZSTR Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
00	None	**Meaning:** IXLZSTR completed successfully and returned the requested information in the data area provided. **Action:** None
04	04	**Meaning:** Not all requested data could be returned in the data area provided because the data area was not large enough. **Action:** To retrieve the remainder of the data, invoke IXLZSTR again with the same keywords and the RESTOKEN as input to the macro.
08	04	**Meaning:** The STRNAME specified on the IXLZSTR macro does not appear in the dump. **Action:** Correct STRNAME, if incorrect, and reissue the IXLZSTR macro.
08	08	**Meaning:** The STRNAME specified on the IXLZSTR macro appears in the dump, but the STRDUMPID specified does not appear in the dump **Action:** Correct STRDUMPID, if incorrect, and reissue the IXLZSTR macro.
08	0C	**Meaning:** No coupling facility data appears in the dump. **Action:** Ensure that you have specified the correct dump.
08	10	**Meaning:** ANSAREA specified on the IXLZSTR macro does not meet the minimum storage requirement for the request. **Action:** Recalculate the amount of storage required for the type of request you are submitting and reissue the IXLZSTR macro.
08	14	**Meaning:** The data does not appear in the dump because the attributes of the requested data does not match the attributes of the structure type. **Action:** Correct the attributes specified and reissue the IXLZSTR macro.
08	18	**Meaning:** The range specification on the IXLZSTR macro is not valid. The starting value is greater than the ending value. **Action:** Correct the range values and reissue the IXLZSTR macro.
0C	04	**Meaning:** Environmental error. Unable to obtain system storage. **Action:** Retry the request one or more times. If the problem persists, record the return and reason codes and supply them to the appropriate IBM support personnel.

Table 88. Return and Reason Codes for the IXLZSTR Macro (continued)

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning and Action
10	04	**Meaning:** Failure in IXLZSTR processing. Some storage could not be obtained in the dump data set. **Action:** Save the return code information and contact the IBM support center.

Example

```
         TITLE  'XETPUB02-Sample IPCS exit using IXLZSTR with RESTOKEN'
*01* FUNCTION =
*          Sample program to illustrate use of IXLZSTR macro
*        in an IPCS environment to access coupling facility
*        structure data from a dump containing that information.  Usage
*        of the RESTOKEN keyword is shown for handling requests for
*        which all of the information can not be returned in the
*        user-provided answer area.
*
*02*   OPERATION =
*         (1) Obtain a 4K (4096 byte) buffer to be passed to the
*             IXLZSTR macro for use as the answer area for a summary
*             type request.
*
*         (2) Issue the IXLZSTR request to obtain summary structure
*             information from the input dump which is expected to
*             contain coupling facility structure data for a cache
*             structure.  The dump also contains multiple entries for
*             a given castout class value and the entries have adjunct
*             and entry data that was also dumped.
*
*         (3) The answer area is searched looking for summary structure
*             information for a specific CACHE structure name that is
*             expected to have been dumped. If it is found then the
*             structure dump id is saved for subsequent use
*             in obtaining class entry information for a given castout
*             class value.
*
*         (4) Free 4K (4096 byte) buffer previously obtained for use as
*             a summary data answer area.
*
*         (5) Obtain a 8K (8192 byte) buffer to be passed to the
*             IXLZSTR macro for use as the answer area for a entry data
*             type request.
*
*         (6) Issue the IXLZSTR request to obtain all entries with both
*             adjunct and entry data returned for a given castout class
*             value for the specific Cache structure.
*
*             As parts of the answer area are "analyzed", WTOs will
*             be issued:
*               - Indicating start of analysis of the answer area
*               - Identifying the entry name from entry controls
*               - Displaying first 16 bytes of adjunct and entry data
*                 for each entry
*               - Indicating going to next table entry in answer area
*               - Indicating finished with answer area
*
*         (7) Process all of the table entries returned in the answer
*             area.  Since TYPE(CLASS) data with adjunct and entry data
*             is requested and size of entry data is quite large,
*             multiple invocations of IXLZSTR are made using the
*             RESTOKEN returned to get more data that can not fit in
*             the returned answer area for the first and subsequent
*             IXLZSTR invocations. (I.e. Step 6 and 7 is repeated as
*             many times as necessary to get all of the data.)
*
*         (8) Free 8K (8192 byte) buffer previously obtained for use as
*             a entry data answer area.
*
*         (9) Free dynamic storage previously obtained and return to
*             caller.
**********************************************************************
*
*02* RECOVERY-OPERATION = This program functions without recovery.
*
**********************************************************************
         EJECT
************************************************************************
*                                                                      *
*   Standard entry linkage                                             *
*                                                                      *
************************************************************************
         STM    R14,R12,12(R13)
         BALR   BASEREG1,0             Establish addressability
         USING  *,BASEREG1
         MODID  BR=YES
         LR     R2,R1                  Save input parameter
         LR     R3,R13                 Save callers savearea address
         STORAGE OBTAIN,ADDR=(DATAREG1),SP=0,LENGTH=DYNASIZE
         LA     DATAREG2,4095(,DATAREG1) Set Second Data Register
         USING  DYNA,DATAREG1          First Data Register
         USING  DYNA+4095,DATAREG2     Second Data Register
         ST     R3,SAVEAREA+4          Save @ of callers savearea
         ST     DATAREG1,8(,R3)        Chain our savearea to callers
         EJECT
************************************************************************
*                                                                      *
*   Initialize variables                                               *
*                                                                      *
************************************************************************
         MVC    EXITRC,=AL4(GOODRETC) * Initialize return code
         ST     R2,ABDPLPTR          Save input pointer to ABDPL
         MVC    WTOEXEC(LENWTOS),WTOS * Copy static parmlist to dynamic
         MVC    WTOTXTD1(L'WTOTXTD1),WTOTXTS1 * Prime WTO text length
         EJECT
************************************************************************
* (1) Obtain a 4K (4096 byte) buffer to be passed to the               *
*     IXLZSTR macro for use as the answer area for a summary           *
*     type request.                                                    *
************************************************************************
         L      R0,SUMMSIZE          Size of summary buffer to obtain
         STORAGE OBTAIN,ADDR=ANSAREA_PTR,SP=0,LENGTH=(R0)
         USING AREAMAP,R2
         L     R2,ANSAREA_PTR        Get addressability to Answer area
         EJECT
************************************************************************
* (2) Issue the IXLZSTR request to obtain summary structure            *
*     information from the input dump which is expected to contain    *
*     Coupling Facility structure data for a Cache structure.  The    *
*     dump should also contain multiple entries for a given castout   *
*     class value and the entries should have adjunct and entry data  *
*     that was also dumped.                                            *
*                                                                      *
************************************************************************
         XC    MYRESTOKEN(64),MYRESTOKEN Initialize RESTOKEN to
*                                    binary zeros
         IXLZSTR ANSAREA=AREAMAP,    Output answer area                +
               ANSLEN=SUMMSIZE,      Answer area length                +
               RESTOKEN=MYRESTOKEN,  Input/Output token for IXLZSTR    +
               ABDPLPTR=ABDPLPTR,    IPCS common parameter List addres +
               TYPE=STRUCTURE,       Get Structure information         +
               STRLEVEL=SUMMARY,     Get summary information           +
               RETCODE=SAVERET,      Return Code                       +
               RSNCODE=SAVERSN,      Reason Code                       +
               MF=(E,STREXEC)
***********************************************************************
*     Check for the requested data being successfully accessed       *
*                                                                     *
***********************************************************************
         L      R3,SAVERET           Get return code
         C      R3,=AL4(STRBRETCODESUCC) * All data accessed ?
         BNE    BADACC               Data was not successfully accessed
         L      R3,SAVERSN           Get reason code
         C      R3,=AL4(STRBRSNCODESUCC) * All data accessed ?
         BNE    BADACC               Data was not successfully accessed
         EJECT
***********************************************************************
*     Summary data was successfully accessed in the dump              *
*                                                                     *
***********************************************************************
* (3) Search the answer area table entries (which contain summary     *
*     structure information for the input Cache structure that this   *
*     sample IPCS exit is interested in.  If the structure is found   *
*     then the corresponding structure dump id is saved for subsequent*
*     use in obtaining class entry information for the input castout  *
*     class value.                                                    *
*                                                                     *
* Note: If the Cache structure of interest was in Structure Rebuild   *
*     there may be more than one dumped instant of the Cache          *
*     structure. For purposes of this sample exit, the installation   *
*     is assumed to be interested in the first dumped instance of the *
*     structure.                                                      *
*                                                                     *
***********************************************************************
         WTO 'XETPUB02 ANALYZING SUMMARY ANSWER AREA',ROUTCDE=11
         USING STRBHEADER,R2         Base Header section of ANSAREA
         USING STRBSTRSUMMARY,R4     Base Structure Summary table entry
         L      R4,STRBFIRSTTABLEENTRY@ * Get addressability to first
*                                    table entry in answer area
         NI     WRKFLG,NOTFOUND      Did not find Cache structure yet
         LA     R6,STRBSTRSUMMARY_LEN * Size of summary table entry
         LA     R8,1                 Loop increment
         LR     R9,R8                Processing first table entry
NEXTENT  ST     R9,I                 Remember loop count
         L      R5,STRBNUMTABLEENTRIES * Get number of table entries
         CR     R9,R5                All table entries searched ?
         BH     DONESRH              Yes, stop searching

***********************************************************************
*        Check to see if this table entry is for the input Cache      *
*        structure                                                    *
***********************************************************************
         CLI    STRBSTRSUMMARYTYPE,STRBSTRTYPECACHE * Cache str type ?
         BNE    TRYNEXT              No, Try next table entry
***********************************************************************
*        Check to see if the structure name matches the input Cache   *
*        structure name                                               *
***********************************************************************
         CLC    STRBSTRSUMMARYNAME,APPSTRNM * Same str name ?
         BNE    TRYNEXT              No, Try next table entry
***********************************************************************
*        Found the table entry for the input Cache structure name.    *
*        (NOTE: More code needed to handle cases where structure is   *
*         dumped more than once -- e.g. Structure in rebuild)         *
***********************************************************************
         OI     WRKFLG,FOUNDIT       Indicate Cache structure found
         LH     R3,STRBSTRSUMMARYSTRDUMPID * Save structure dump id
         STH    R3,DUMPID_STR        Remember structure dump id
         B      DONESRH              Stop searching for structure
TRYNEXT  EQU    *                    Try next table entry (if any)
         ALR    R4,R6                Point to next table entry
         ST     R4,SUMMARY_PTR       Remember new table entry address
         ALR    R9,R8                Processing another table entry
         B      NEXTENT              Go process the next table entry
         SPACE  1
***********************************************************************
*     Finished searching returned structure summary information      *
***********************************************************************
DONESRH  EQU    *                    Finished search for structure
         TM     WRKFLG,FOUNDIT       Found the input Cache structure ?
         BZ     NOCACHE              No, tell user structure not found
         WTO 'XETPUB02 CACHE STRUCTURE FOUND IN DUMP',ROUTCDE=11
         B      FREESUM              Go free answer area buffer
NOCACHE  EQU    *                    Did not find the Cache structure
         WTO 'XETPUB02 CACHE STRUCTURE NOT FOUND IN DUMP',ROUTCDE=11
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         B      FREESUM              Go free answer area buffer
         EJECT
***********************************************************************
*     Summary data was not successfully accessed in the dump         *
*                                                                     *
***********************************************************************
BADACC   EQU    *                    Data was not accessed successfully
         MVC    WTOTXTD2(L'WTOTXTD2),WTOBADAC * Set message text
         MVC    MAPSERV(8),=CL8'IXLZSTR ' * Service that failed
* Convert hex return code to printable hex
         MVC    PHEXIN(4),SAVERET    Hex return code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRETC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRETC(8),TRTBL-240 Translate to printable hex
* Convert hex reason code to printable hex
         MVC    PHEXIN(4),SAVERSN    Hex reason code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRSNC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRSNC(8),TRTBL-240 Translate to printable hex
         BAL    R14,ISSUEWTO         Tell user access failed
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         EJECT
***********************************************************************
* (4) Free 4K (4096 byte) buffer previously obtained for a summary   *
*     data answer area.                                               *
*                                                                     *
***********************************************************************
FREESUM  L      R0,SUMMSIZE          Size of summary buffer to be freed
         STORAGE RELEASE,ADDR=((R2)),SP=0,LENGTH=(R0)
         DROP   R2
***********************************************************************
*     Only continue processing if input cache structure was dumped   *
***********************************************************************
         TM     WRKFLG,FOUNDIT       Found the input Cache structure ?
         BZ     COMPLETE             No, exit is finished
         SPACE 2
***********************************************************************
* (5) Obtain a 8K (8192 byte) buffer to be passed to the             *
*     IXLZSTR macro for use as the answer area for a Entry data       *
*     type request.                                                   *
*                                                                     *
***********************************************************************
         L      R0,BIGSIZE           Size of big buffer to be obtained
         STORAGE OBTAIN,ADDR=ANSAREA_PTR,SP=0,LENGTH=(R0)
         USING AREAMAP,R2
         L     R2,ANSAREA_PTR        Get addressability to Answer area
         EJECT
```

```
***********************************************************************
* (6) Issue the IXLZSTR request to obtain all entries with both       *
*     adjunct and entry data returned for a given input castout class *
*     value for the input Cache structure.                            *
*                                                                     *
*     This exit is interested in listing the first 16 bytes of the    *
*     entry data and adjunct data for each entry. For illustration    *
*     purposes a WTO will be issued to job log for each matching      *
*     entry found displaying this data.                               *
*                                                                     *
*     The expected amount of data that will be returned in the answer *
*     area is quite large and will not fit in the coded 8K answer     *
*     area buffer.  The following code illustrates how to use         *
*     IXLZSTR macro with RESTOKEN keyword in a loop to retrieve all   *
*     of the requested data.                                          *
*                                                                     *
***********************************************************************
         LA    R5,ENTBUF             Point to start of entry data WTO
*                                    buffer
         ST    R5,ENTBUF_PTR         Remember location in buffer
         L     R5,MIN_ENTDATA_SIZE   Get size of buffer
         ST    R5,LEN_NEEDED         Remember size of buffer still to
*                                    be copied
         NI    WRKFLG,NOENTWTO       Remember that WTO has not been
*                                    issued for current table entry
         XC    MYRESTOKEN(64),MYRESTOKEN Initialize RESTOKEN to
*                                    binary zeros
***********************************************************************
*     Continue to request more of the entry data until all of the     *
*     data has been returned. (Start of loop)                         *
*                                                                     *
***********************************************************************
GETDATA  IXLZSTR ANSAREA=AREAMAP,    Output answer area               +
               ANSLEN=BIGSIZE,       Answer area length               +
               RESTOKEN=MYRESTOKEN,  Input/Output token for IXLZSTR   +
               ABDPLPTR=ABDPLPTR,    IPCS common parameter List addr  +
               STRNAME=APPSTRNM,     Structure name to get info on    +
               STRDUMPID=DUMPID_STR, Structure dump id to get info on +
               TYPE=CLASS,           Get Structure class information  +
               CLASSTYPE=CASTOUT,    Get castout class information    +
               CLASSLEVEL=ENTRY,     Get entry information            +
               CLASSVAL=GETCLASS,    Class value to get info on       +
               ADJUNCT=YES,          Request adjunct data             +
               ENTRYDATA=YES,        Request entry data               +
               ORDER=TAIL,           Return data in tail-to-head order +
               RETCODE=SAVERET,      Return Code                      +
               RSNCODE=SAVERSN,      Reason Code                      +
               MF=(E,STREXEC)
         SPACE  1
***********************************************************************
*     Check for the requested data being successfully accessed        *
***********************************************************************
         L      R3,SAVERET           Get return code
         C      R3,=AL4(STRBRETCODESUCC) * All data accessed ?
         BNE    CHKSOME              No, Only part of data accessed ?
         L      R3,SAVERSN           Get reason code
         C      R3,=AL4(STRBRSNCODESUCC) * All data accessed ?
         BNE    BADACC2              Data was not successfully accessed
         B      GOODACC              Data was successfully accessed
CHKSOME  C      R3,=AL4(STRBRETCODEMOREDATA) * More data to get ?
         BNE    BADACC2              Data was not successfully accessed
         L      R3,SAVERSN           Get reason code
         C      R3,=AL4(STRBRSNCODEANSANOTLGE) * Answer area too small?
         BNE    BADACC2              Data was not successfully accessed
         EJECT
***********************************************************************
* (7) Process all of the table entries returned in the answer area    *
*     from the IXLZSTR invocation.  Since TYPE(CLASS) was requested   *
*     with adjunct and entry data also returned for a given castout   *
*     class value and the size of the entry data is quite large, the  *
*     contents of the answer area just returned may contain different *
*     pieces of information.                                          *
*                                                                     *
*     The first thing that appears in the answer area is always the   *
*     answer area header section mapped by the IXLZSTRB mapping macro *
*     (section STRBHEADER).  The header section will point to the     *
*     first table entry returned in the current answer area.          *
*                                                                     *
*     The format of each table entry returned for a TYPE(CLASS)       *
*     CLASSLEVEL(ENTRY) request is also mapping by IXLZSTRB in the    *
*     mapping section beginning with field STRBENTRY.                 *
*                                                                     *
***********************************************************************
GOODACC  EQU   *                     Data was successfully accessed
         WTO 'XETPUB02 ANALYZING ANSWER AREA',ROUTCDE=11
         USING STRBHEADER,R2         Base Header section of ANSAREA
         USING STRBENTRY,R4          Base TYPE(CLASS) CLASSLEVEL(ENTRY)
*                                    table entry
         L      R4,STRBFIRSTTABLEENTRY@ * Get addressability to first
*                                    table entry in answer area
         LA     R8,1                 Loop increment
         LR     R9,R8                Processing first table entry
NEXTENT2 ST     R9,I                 Remember number of table entries
*                                    processed
***********************************************************************
*     The STRBENTRY will indicate if the current table entry has any  *
*     entry controls present, where it is located and its size. The   *
*     presence, location, and size of any adjunct data is also in the *
*     STRBENTRY.  Also, the presence, location, and size of any entry *
*     data will be indicated in the STRBENTRY information.            *
*                                                                     *
***********************************************************************
         SPACE  1
***********************************************************************
*     The first returned answer area associated with a given table    *
*     entry will point to any entry controls and adjunct data that is *
*     going to be returned for the entry.                             *
***********************************************************************
         EJECT
***********************************************************************
*     Check for entry control data                                    *
***********************************************************************
         L      R3,STRBENTRYCNTL@    Get pointer to entry control data
         C      R3,ZERO              Entry control data present ?
         BZ     SKIPCNTL             No, Skip entry control processing
***********************************************************************
*     Process the entry controls associated with the current table    *
*     entry.  The entry controls for a cache structure are mapped by  *
*     DDIC mapping in mapping macro IXLYDDIB.                         *
*                                                                     *
*     The code below will access the DDIC to get the value specified  *
*     for the entry when the data was registered in the cache.  This  *
*     information is surfaced by a WTO to the job log.                *
***********************************************************************
         USING  DDIC,R3              Base the cache entry controls
         MVC    WTOTXTD2(L'WTOTXTD2),WTOCNT * Set message text
         MVC    MAPCNTNM,DDICNAME    Copy the entry's name
         BAL    R14,ISSUEWTO         Issue WTO
SKIPCNTL EQU    *                    Label skips entry control code
         SPACE  1
***********************************************************************
*     Check for adjunct data                                          *
***********************************************************************
```

```
         L      R3,STRBENTRYADJ@     Get pointer to adjunct data
         C      R3,ZERO              Adjunct data present ?
         BZ     SKIPADJ              No, Skip adjunct data processing
***********************************************************************
*     Process the adjunct data associated with the current table      *
*     entry.  Adjunct data is application specific --- the following  *
*     section of code just issues a WTO to show the first 16 bytes.   *
***********************************************************************
         USING  ADJDATA,R3           Base adjunct data area
         MVC    WTOTXTD2(L'WTOTXTD2),WTOADJ * Set message text
         MVC    MAPADJ,ADJ16         First 16 bytes of adjunct data
         BAL    R14,ISSUEWTO         Issue WTO
SKIPADJ  EQU    *                    Label skips adjunct data code
         EJECT
***********************************************************************
*     Process the entry data associated with the current table        *
*     entry.  Entry data is application specific --- the following    *
*     section of code just issues a WTO to show the first 16 bytes    *
*     (see constant MIN_ENTDATA_SIZE).                                *
*                                                                     *
*     NOTE: This code assumes that entry data for the structure is    *
*     always at least 16 bytes.  If a entry is found with less entry  *
*     data than the expected application minimum size, then a WTO is  *
*     issued (which would have been proceeded by a WTO that showed    *
*     the object name associated with the entry from entry controls). *
*                                                                     *
*     There may or may not be room for a table entry's entry data in  *
*     the first answer area returned for the entry.  Entry data may   *
*     also span one or more answer areas. The code below accumulates  *
*     entry data in a WTO buffer until MIN_ENTDATA_SIZE bytes of      *
*     entry data have been returned.  Even though there may be more   *
*     entry data than MIN_ENTDATA_SIZE only MIN_ENTDATA_SIZE bytes    *
*     are saved for purposes of the WTO showing the first part of the *
*     entry data.                                                     *
*                                                                     *
***********************************************************************
*                                                                     *
*        NOTE: Number of IXLZSTR invocations needed to get all of the *
*     entry data for a given table entry can be minimized by taking   *
*     advantage of the STRBENTRYEDATALENLEFT2PROC field to            *
*     determine how big of an answer area to provide to IXLZSTR.      *
***********************************************************************
         TM     WRKFLG,ENTWTO        Processed entry data for this
*                                    table entry yet ?
         BNZ    SKIPENT              Yes, Skip Entry data processing
         L      R7,STRBENTRYTOTALEDATALEN Get total entry data length
*                                    associated with the table entry
         C      R7,MIN_ENTDATA_SIZE    Total entry data size is large
*                                    enough ?
         BNL    OKTOTSZ              Yes, Process entry data
***********************************************************************
*     Issue WTO indicating that the entry data for a table entry was  *
*     too small. Previous WTO may have already been issued with       *
*     object name associated with this table entry.                   *
***********************************************************************
         MVC    WTOTXTD2(L'WTOTXTD2),WTOBADSZ * Set message text
         BAL    R14,ISSUEWTO         Issue WTO
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         OI     WRKFLG,ENTWTO        Indicate WTO issued about this
*                                    table entry's entry data
         B      SKIPENT              Skip Entry data processing
*                                    table entry's entry data
         EJECT
OKTOTSZ  EQU    *                    Label total entry data size OK
         L      R3,STRBENTRYEDATA@   Get pointer to entry data
         C      R3,ZERO              Entry data present ?
         BZ     SKIPENT              No, Skip Entry data processing
*                                    associated with the table entry
***********************************************************************
*     Some entry data is present in this answer area for the current  *
*     table entry being processed                                     *
***********************************************************************
         L      R5,STRBENTRYEDATALEN Get amount of entry data returned
         L      R6,LEN_NEEDED        Get amount of entry data still
*                                    needed before entry data WTO can
*                                    be issued
         CLR    R5,R6                Was enough entry data returned to
*                                    issue the entry data WTO ?
         BL     MOVEPART             No, move the part of entry data
*                                    in this answer area to WTO buffer
         SPACE 1
***********************************************************************
*     Enough entry data is present in this answer area to fill up the *
*     WTO buffer and issue the entry data WTO.                        *
***********************************************************************
         L      R7,ENTBUF_PTR        Point to where to move data to
         BCTR   R6,0                 Set length of data to move
         EX     R6,@MOVEENT          Move entry data to WTO buffer
         MVC    WTOTXTD2(L'WTOTXTD2),WTOENT * Set message text
         MVC    MAPENT,ENTBUF        Move entry data into WTO
         BAL    R14,ISSUEWTO         Issue WTO
         OI     WRKFLG,ENTWTO        Indicate WTO issued about this
*                                    table entry's entry data
         B      SKIPENT
MOVEPART EQU    *                    Label to move only part of needed
*                                    entry data to WTO buffer
         SLR    R6,R5                Calculate amount of entry data
*                                    still needed for this table entry
         ST     R6,LEN_NEEDED        Remember how much is still needed
         L      R7,ENTBUF_PTR        Point to where to move data to
         BCTR   R5,0                 Get size of data to be moved
         EX     R5,@MOVEENT          Move entry data to WTO buffer
         ALR    R7,R5                Point to next byte to move entry
         ALR    R7,R8                Add one to adjust for prior BCTR
         ST     R7,ENTBUF_PTR        Remember next byte in WTO buffer
*                                    to move entry data to
         EJECT
SKIPENT  EQU    *                    Skip entry data processing for
*                                    current table entry
***********************************************************************
*     Determine if all of the entry data for current table entry has  *
*     been seen.                                                      *
***********************************************************************
         L      R5,STRBENTRYEDATALENLEFT2PROC * Get amount of entry
*                                    data left to process for last
*                                    table entry in this answer area
         C      R5,ZERO              More entry data for this entry ?
         BNE    CHKLAST              Yes, check that it is last one

***********************************************************************
*     All of the entry data for this table entry has been seen.       *
*     Reset entry data buffer indicators for next table entry.        *
***********************************************************************
         LA    R5,ENTBUF             Point to start of entry data WTO
*                                    buffer
         ST    R5,ENTBUF_PTR         Remember location in buffer
         L     R5,MIN_ENTDATA_SIZE   Get size of buffer
         ST    R5,LEN_NEEDED         Remember size of buffer still to
*                                    be copied
         NI    WRKFLG,NOENTWTO       Remember that WTO has not been
*                                    issued for current table entry
CHKLAST  EQU    *                    Check for last table entry
***********************************************************************
*        Determine if there is another table entry to process in the  *
*       current answer area                                           *
***********************************************************************
         L      R9,I                 Get number of entries processed
         ALR    R9,R8                Increment number of table entries
         L      R5,STRBNUMTABLEENTRIES * Get number of table entries
         CR     R9,R5                All table entries processed ?
         BH     DONEENT              Yes, stop processing table entries
         WTO 'XETPUB02 GOING TO NEXT TABLE ENTRY',ROUTCDE=11
         EJECT
***********************************************************************
*       Point to the next table entry. The following calculation     *
*       will always give you the length to add to get to the next    *
*       table entry:                                                  *
*                                                                     *
* size table entry +size adjunct   +size entry data +size entry cntls *
*              (i.e.)                                                 *
* STRBTABLEENTRYLEN+STRBENTRYADJLEN+STRBENTRYEDATLEN+STRBENTRYCNTLLEN *
*                                                                     *
*       NOTE: Some of the sizes above can be different for each      *
*             table entry and different for the "same table entry"   *
*             for a table entry for which not all of the entry data  *
*             can fit in the provided answer area in one IXLZSTR     *
*             macro invocation.                                       *
*                                                                     *
***********************************************************************
         L      R5,STRBTABLEENTRYLEN * Get size of table entry
         L      R6,STRBENTRYADJLEN   Get size of returned adjunct data
         ALR    R5,R6                Add to table entry size
         L      R6,STRBENTRYEDATALEN Get size of returned entry data
         ALR    R5,R6                Add to table entry size
         L      R6,STRBENTRYCNTLLEN  Get size of returned entry cntls
         ALR    R5,R6                Add to table entry size
         ALR    R4,R5                Point to next table entry
         ST     R4,ENTRY_PTR         Remember new table entry address
         B      NEXTENT2             Go process the next table entry
         EJECT
***********************************************************************
*     Finished processing returned TYPE(CLASS) CLASSLEVEL(ENTRY) data *
*     in the current answer area buffer.                              *
***********************************************************************
DONEENT  EQU    *                    Finished processing table entries
         WTO 'XETPUB02 FINISHED WITH THIS ANSAREA',ROUTCDE=11

***********************************************************************
*     Check to see if there is more data still to get from dump      *
***********************************************************************
         L      R3,SAVERET           Is there more data to get ?
         C      R3,=AL4(STRBRETCODEMOREDATA) * More data to get ?
         BE     GETDATA              Go get more of the data
***********************************************************************
*     (End of loop to request more of the entry data in the dump     *
*     until of the data has been returned)                            *
*                                                                     *
***********************************************************************
         WTO 'XETPUB02 FINISHED ACCESSING ALL DATA',ROUTCDE=11
         B      FREEENT              Go free answer area buffer
         EJECT
***********************************************************************
*     Entry data was not successfully accessed in the dump           *
*                                                                     *
***********************************************************************
BADACC2  EQU    *                    Data was not accessed successfully
         MVC    WTOTXTD2(L'WTOTXTD2),WTOBADAC * Set message text
         MVC    MAPSERV(8),=CL8'IXLZSTR ' * Service that failed
* Convert hex return code to printable hex
         MVC    PHEXIN(4),SAVERET    Hex return code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRETC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRETC(8),TRTBL-240 Translate to printable hex
* Convert hex reason code to printable hex
         MVC    PHEXIN(4),SAVERSN    Hex reason code to convert
         UNPK   PHEXOUT,PHEXIN       Unpack the data
         MVC    MAPRSNC(8),PHEXOUT+1 Store unpacked data into target
         TR     MAPRSNC(8),TRTBL-240 Translate to printable hex
         BAL    R14,ISSUEWTO         Tell user access failed
         MVC    EXITRC,=AL4(BADRETC) Set bad return code
         EJECT
***********************************************************************
* (8) Free 8K (8192 byte) buffer previously obtained for a entry      *
*     data answer area.                                               *
*                                                                     *
***********************************************************************
FREEENT  L      R0,BIGSIZE           Size of entry buffer to be freed
         STORAGE RELEASE,ADDR=((R2)),SP=0,LENGTH=(R0)
         DROP   R2
         EJECT
***********************************************************************
*                                                                     *
* (9) Free up dynamic storage and return to caller                    *
*                                                                     *
***********************************************************************
COMPLETE EQU    *
         L      R2,SAVEAREA+4        Save caller's save area address
         L      R3,EXITRC            Save IPCS exit return code
         STORAGE RELEASE,ADDR=((DATAREG1)),LENGTH=DYNASIZE
         LR     R13,R2               Restore caller's save area address
         L      R14,12(R13)          Restore Return address
         LR     R15,R3               Set IPCS exit return code
         LM     R0,R12,20(R13)       Restore Registers R0-R12
         BR     R14                  Return to caller
         SPACE  2
***********************************************************************
*     Special EX target instructions                                  *
***********************************************************************
@MOVEENT MVC    0(0,R7),0(R3)          Move entry data to WTO buffer
         EJECT
***********************************************************************
*                                                                     *
*   Subroutine: ISSUEWTO                                              *
*                                                                     *
*   Function  : This routine is called whenever contention is         *
*               detected on a dataset, and the dataset owner is a     *
*               job on this system. It will take whatever action it   *
*               can to attempt to relieve the contention.             *
*                                                                     *
*   Input     : WTOTXTD1 contains text for WTO message to be issued   *
*                                                                     *
*                                                                     *
***********************************************************************
ISSUEWTO EQU *
         STM    R14,R12,SAVE1          Save callers regs
         LA     R5,WTOTXTD1            Address WTO parmlist
         WTO TEXT=(R5),ROUTCDE=(11),MF=(E,WTOEXEC) * Issue WTO
         LM     R14,R12,SAVE1          Restore callers regs
         BR     R14                    Return to caller
         EJECT
***********************************************************************
*                                                                     *
*   Register declares                                                 *
*                                                                     *
***********************************************************************
R0       EQU    0
R1       EQU    1
```

```
R2       EQU    2
R3       EQU    3
R4       EQU    4
R5       EQU    5
R6       EQU    6
R7       EQU    7
R8       EQU    8
R9       EQU    9
R10      EQU    10                    Reserved for future expansion
*                                     of the code or the dynamic area
DATAREG2 EQU    11                    Second data register
BASEREG1 EQU    12                    Code register
R12      EQU    12
DATAREG1 EQU    13                    First data register
R13      EQU    13
R14      EQU    14
R15      EQU    15
         EJECT
***********************************************************************
*                                                                     *
*   Static data                                                       *
*                                                                     *
***********************************************************************
         DS     0F
TRTBL    DC     CL16'0123456789ABCDEF' * Translate table
ZERO     DC     F'0'                  * Constant zero for comparisons
         SPACE  1
***********************************************************************
*                                                                     *
*   Static WTO data                                                   *
*                                                                     *
***********************************************************************
WTOS     WTO TEXT=,ROUTCDE=(11),MF=L * Static form of WTO
LENWTOS  EQU *-WTOS                  * Length of WTO parmlist
WTOTXTS1 DC     AL2(L'WTOTXTD2)      * WTO text length
WTOBADAC DC CL65'XETPUB02 mmmmmmmm RETCODE=rrrrrrrr RSNCODE=ssssssss'
MAPSERV  EQU    WTOTXTD2+9,8,C'C'    * Map service WTO insert
MAPRETC  EQU    WTOTXTD2+26,8,C'C'   * Map service RETCODE insert
MAPRSNC  EQU    WTOTXTD2+43,8,C'C'   * Map service RETCODE insert
WTOADJ   DC CL65'XETPUB02 FIRST 16 CHARS ADJUNCT IS: aaaaaaaaaaaaaaaa'
MAPADJ   EQU    WTOTXTD2+36,16,C'C'   * Map adjunct data insert
WTOENT   DC CL65'XETPUB02 FIRST 16 CHARS ENTRY DATA: eeeeeeeeeeeeeeee'
MAPENT   EQU    WTOTXTD2+36,16,C'C'   * Map entry data insert
WTOBADSZ DC CL65'XETPUB02 ENTRY DATA SIZE WAS TOO SMALL              '
WTOCNT   DC CL65'XETPUB02 ENTRY NAME FROM DDICNAME:  cccccccccccccccc'
MAPCNTNM EQU    WTOTXTD2+36,16,C'C'   * Map entry's object name insert
         EJECT
***********************************************************************
*                                                                     *
*   Constants used in accessing Coupling Facility structure data.     *
*                                                                     *
***********************************************************************
APPSTRNM DC     CL16'CACHE02         ' Application structure name to
*                                     be accessed in the dump
GETCLASS DC     F'2'                  Castout class value for which
*                                     class entry data is to be
*                                     accessed
SUMMSIZE DC     F'4096'               Size of answer area to be used
*                                     for returning coupling facility
*                                     summary information
BIGSIZE  DC     F'8192'               Size of answer area to be used
*                                     for returning coupling facility
*                                     detail information
MIN_ENTDATA_SIZE DC F'16'             Minimum size of entry data
*                                     expected. If this size is
*                                     changed then WTO sizes may have
*                                     to be updated.
*                                      (See ENTBUF, WTOENT, MAPENT, and
*                                       WTOTXTD2 fields)
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*   Dynamic data                                                      *
*                                                                     *
***********************************************************************
DYNA     DSECT
SAVEAREA DS     18F                    Standard savearea (first field)
ABDPLPTR DS     AL4                    Pointer to ABDPL
ANSAREA_PTR DS  AL4                    Answer area storage pointer
LEN_NEEDED  DS  F                      Length of entry data still
*                                      needed before WTO is issued
CLASSVAL DS     1F                     Class value
DETAIL_PTR  DS  A                      ANSAREA detail area data pointer
DUMPID_STR  DS  1H                     Structure dump id
DUMP_STRNAME DS CL16                   Structure name in dump
ENTBUF      DS CL16                    Entry data buffer for WTO
            DS  0F
ENTBUF_PTR  DS  A                      Pointer into entry data buffer
*                                      to copy entry data into
ENTRY_PTR   DS  A                      ANSAREA Entry area pointer
I        DS     1F                     Loop index
PHEXIN   DS     CL5                    Work area for printable hex conv
PHEXOUT  DS     CL10                   Work area for printable hex conv
MYRESTOKEN DS   CL64                   RESTOKEN returned by IXLZSTR
SAVERET  DS     F                      Save macro service return code
SAVERSN  DS     F                      Save macro service reason code
SAVE1    DS     15F                    First level subroutine savearea
SAVE2    DS     15F                    Second level subroutine savearea
SUMMARY_PTR DS  A                      ANSAREA summary data pointer
EXITRC   DS     F                      Module Return code
BADRETC  EQU    8                      Bad return code from IPCS exit
GOODRETC EQU    0                      Good return code from IPCS exit
         SPACE  1
WRKFLG   DS     BL.008                 Work Flags
FOUNDIT  EQU    X'80'                  Indicates Input Cache structure
*                                      summary info found in the dump
ENTWTO   EQU    X'40'                  Indicates WTO issued about entry
*                                      data for current entry
NOENTWTO EQU    X'BF'                  Indicates WTO not issued about
*                                      entry data for current entry
NOTFOUND EQU    X'7F'                  Indicates summary info not found
*                                      in the dump
         EJECT
***********************************************************************
*                                                                     *
*   List forms of macros (Dynamic storage)                            *
*                                                                     *
***********************************************************************
         IXLZSTR MF=(L,STREXEC)
         EJECT
***********************************************************************
*                                                                     *
*   Dynamic WTO storage                                               *
*                                                                     *
***********************************************************************
WTOEXEC  WTO TEXT=,ROUTCDE=(11),MF=L * List form of WTO parmlist
WTOTXTD1 DC     AL2(L'WTOTXTD1)       * WTO text length
WTOTXTD2 DC     CL65' '               * WTO text
***********************************************************************
*                                                                     *
*   End of dynamic storage                                            *
*                                                                     *
```

```
***********************************************************************
DYNASIZE EQU    *-DYNA                 Total size of dynamic storage
         EJECT
***********************************************************************
*                                                                     *
*   Other mappings                                                    *
*                                                                     *
***********************************************************************
AREAMAP  EQU    0,,C'C'                Map Answer area
         SPACE 2
ADJDATA  DSECT                         Map Adjunct data
ADJ16    DS CL16                       First 16 bytes adjunct data
ADJREST  DS CL48                       Rest of adjunct data
         EJECT
***********************************************************************
*                                                                     *
*   Mapping macros                                                    *
*                                                                     *
***********************************************************************
         BLSABDPL
         IXLZSTRB
         IXLYDDIB
         END
```

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User’s Guide*, and *z/OS ISPF User’s Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain services of z/OS.

Trademarks

Index

Readers' Comments — We'd Like to Hear from You

z/OS
MVS Programming:
Sysplex Services Reference

Publication No. SA22-7618-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SA22-7618-03

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

SA22-7618-03

Cut or Fold
Along Line

Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

SA22-7618-03